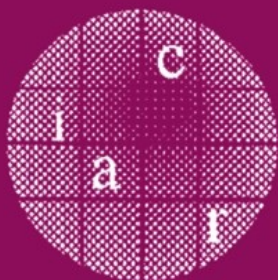


Yuliang Zheng (Ed.)

LNCS 2501

# Advances in Cryptology – ASIACRYPT 2002

8th International Conference on the Theory  
and Application of Cryptology and Information Security  
Queenstown, New Zealand, December 2002  
Proceedings



Springer

# Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

2501

**Springer**

*Berlin*

*Heidelberg*

*New York*

*Barcelona*

*Hong Kong*

*London*

*Milan*

*Paris*

*Tokyo*

Yuliang Zheng (Ed.)

# Advances in Cryptology – ASIACRYPT 2002

8th International Conference on the Theory  
and Application of Cryptology and Information Security  
Queenstown, New Zealand, December 1-5, 2002  
Proceedings



Springer

## Series Editors

Gerhard Goos, Karlsruhe University, Germany  
Juris Hartmanis, Cornell University, NY, USA  
Jan van Leeuwen, Utrecht University, The Netherlands

## Volume Editor

Yuliang Zheng  
University of North Carolina at Charlotte  
Department of Software and Information Systems  
9201 University City Blvd, Charlotte, NC 28223, USA  
E-mail: yzheng@uncc.edu

## Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek  
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;  
detailed bibliographic data is available in the Internet at <<http://dnb.ddb.de>>.

CR Subject Classification (1998): E.3, D.4.6, K.6.5, F.2.1-2, C.2, J.1, G.2.2

ISSN 0302-9743

ISBN 3-540-00171-9 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York  
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2002  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Steingräber Satztechnik GmbH, Heidelberg  
Printed on acid-free paper      SPIN: 10870805      06/3142      5 4 3 2 1 0

# Preface

ASIACRYPT 2002 was held in Queenstown, New Zealand, December 1–5, 2002. The conference was organized by the International Association for Cryptologic Research (IACR).

The program committee received 173 submissions from around the world, from which 34 were selected for presentation. Each submission was reviewed by at least three experts in the relevant research area.

Let me first thank all the authors, including those whose submissions were not successful, for taking the time to prepare the submissions. Their dedication and efforts in advancing research in cryptography made this conference possible.

Selecting presentations from such a large number of submissions was an extremely difficult and challenging task. The program committee members, together with external referees, spent thousands of hours of their precious time reviewing the submissions. At the completion of the selection process, the program committee received 875 review reports in total. In addition, the committee received several hundred comments during the three-week period of discussions.

Taking this opportunity, I would like to thank all the program committee members for their time and dedication. Without their expertise in the state of the art in cryptography and their willingness to serve the data security community, the conference would not have had such a high-quality program. I would also like to thank the numerous external referees for their invaluable assistance in identifying the scientific and practical merits of the submissions.

The quality of the program was further enhanced by two distinguished keynote speeches delivered by Prof. Tsutomu Matsumoto from Yokohama National University in Japan, and Dr. Moti Yung from CertCo and Columbia University in the USA. On behalf of the program committee, I would like to thank both prominent pioneers in cryptography for their inspiring presentations.

Thanks also go to the general chair Hank Wolfe from the University of Otago for successfully running the conference in such a beautiful town. It was a wonderful experience for me to work with Hank.

The reviewing process benefited greatly from the advice of Bart Preneel and Wim Moreau on handling the reviewing software. I appreciated Colin Boyd's assistance in editing the proceedings. My special thanks go to Lawrence Teo who acted as my assistant during the entire period of setting up the website, accepting, reviewing submissions, and editing the final proceedings. The year-long process would not have run so smoothly without his tireless help and superb technical skills in handling the software packages.

# ASIACRYPT 2002

December 1–5, 2002, Queenstown, New Zealand

Sponsored by the  
*International Association for Cryptologic Research (IACR)*

## General Chair

Henry Wolfe, University of Otago, New Zealand

## Program Chair

Yuliang Zheng, University of North Carolina at Charlotte, USA

## Program Committee

Feng Bao	LIT, Singapore
Ed Dawson	QUT, Australia
Giovanni DiCrescenzo	Telcordia, USA
Matthew Franklin	UC Davis, USA
Dieter Gollmann	Microsoft Research, UK
Helena Handschuh	Gemplus, France
Philip Hawkes	Qualcomm, Australia
Ari Juels	RSA Laboratories, USA
Kwangjo Kim	ICU, South Korea
Seungjoo Kim	KISA, South Korea
Chi Sung Lai	National Cheng Kung University, Taiwan
Pil Joong Lee	POSTECH, South Korea
Arjen Lenstra	Citibank, USA
Phil MacKenzie	Lucent Technologies, USA
Masahirom Mambo	Tohoku University, Japan
Wenbo Mao	HP Labs, UK
Keith Martin	Royal Holloway, University of London, UK
Alfred Menezes	University of Waterloo, Canada
Phong Nguyen	ENS, France
Dingyi Pei	Chinese Academy of Sciences, China
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Kouichi Sakurai	Kyushu University, Japan
Jessica Staddon	PARC, USA
Serge Vaudenay	EPFL, Switzerland
Sung-Ming Yen	National Central University, Taiwan
Xian-Mo Zhang	University of Wollongong, Australia
Yuliang Zheng (Chair)	UNC Charlotte, USA
Hong Zhu	Fudan University, China

## Advisory Member:

Colin Boyd (Asiacrypt 2001 Program Chair) ..... QUT, Australia

## External Reviewers

Masayuki Abe  
Giuseppe Ateniese  
Gildas Avoine  
Joonsang Baek  
Dirk Balfanz  
Mark Bauer  
Peter Beelen  
Alex Biryukov  
Simon Blackburn  
Daniel Bleichenbacher  
Alexandra Boldyreva  
Dan Boneh  
Colin Boyd  
Emmanuel Bresson  
Eric Brier  
Linda Burnett  
Brice Canvel  
Dario Catalano  
Stefania Cavallar  
Geng Hau Chang  
Chien-ning Chen  
Chien Yuan Chen  
Liqun Chen  
Jung Hee Cheon  
J.H. Chiu  
YoungJu Choie  
Andrew Clark  
Scott Contini  
Jean-Sébastien Coron  
Nicolas Courtois  
Christophe De Cannière  
Alex Dent  
Anand Desai  
Markus Dichtl  
Hiroshi Doi  
Glenn Durfee  
Chun I Fan  
Marc Fischlin  
Yair Frankel  
Martin Gagne  
Steven Galbraith  
Juan Garay  
Katharina Geissler  
Rosario Gennaro

Craig Gentry  
David Goldberg  
Juan Manuel Gonzalez-Nieto  
Louis Goubin  
Louis Granboulan  
Richard Graveman  
Dan Greene  
D.J. Guan  
Jae-Cheol Ha  
Stuart Haber  
Satoshi Hada  
Goichiro Hanaoka  
Darrel Hankerson  
Matthew Henricksen  
Florian Hess  
Shoichi Hirose  
Dennis Hofheinz  
Herbie Hopkins  
Min-Shiang Hwang  
Yong Ho Hwang  
Hisashi Inoue  
Toshiya Itoh  
Tetsu Iwata  
Markus Jakobsson  
Jinn-Ke Jan  
Rob Johnson  
Marc Joye  
Wen-Sheng Juang  
Pascal Junod  
Burt Kaliski  
Masayuki Kanda  
Jonathan Katz  
Alexander Kholosha  
Aggelos Kiayias  
Hiroaki Kikuchi  
Chong Hee Kim  
Neal Koblitz  
Takeshi Koshihara  
Kaoru Kurosawa  
Hidenori Kuwakado  
Tanja Lange  
Dong-Hoon Lee  
Narn-Yih Lee  
Sangjin Lee



Y.C. Lee  
Hsi-Chung Lin  
Chi-Jen Lu  
Chun-Shien Lu  
Christoph Ludwig  
David M'Raihi  
Mike Malkin  
Tal Malkin  
John Malone-Lee  
Takashi Mano  
James McKee  
Bill Millan  
Sara Miner  
Takaaki Mizuki  
Jean Monnerat  
Shiho Moriai  
Siguna Muller  
Bill Munro  
David Naccache  
Koh-ichi Nagao  
Toru Nakanishi  
Kazuo Ohta  
Kazuomi Oishi  
Satomi Okazaki  
Rafail Ostrovsky  
Akira Otsuka  
Pascal Paillier  
Dong Jin Park  
Ji-Hwan Park  
Kenny Paterson  
Giuseppe Persiano  
John Proos  
Michael Quisquater  
Arash Reyhani-Masoleh  
Vincent Rijmen  
Matt Robshaw  
Peter de Rooij  
Greg Rose  
Ludovic Rousseau  
Taiichi Saito  
Ryuichi Sakai  
Jasper Scholten  
Chaofeng Sha

Junji Shikata  
Atsushi Shimbo  
Igor Shparlinski  
Francesco Sica  
Alice Silverberg  
Joe Silverman  
Sang Gyoo Sim  
Leonie Simpson  
Nigel Smart  
Diana Smetters  
David Soldera  
Martijn Stam  
Makoto Sugita  
Hung-Min Sun  
Koutarou Suzuki  
Mike Szydlo  
Mitsuru Tada  
Tsuyoshi Takagi  
Katsuyuki Takashima  
Edlyn Teske  
Yiannis Tsiounis  
Christophe Tymen  
Wen-Guey Tzeng  
Masashi Une  
Frederik Vercauteren  
Eric Verheul  
Kapali Viswanathan  
Jose Vivas  
Huaxiong Wang  
Peter Wild  
Hao-Chi Wong  
Tzong-Chen Wu  
Masato Yamamichi  
Akihiro Yamamura  
Jeff Yan  
Ching-Nung Yang  
Yi-Shiung Yeh  
Yiqun Lisa Yin  
Maki Yoshida  
Dae Hyun Yum  
Fangguo Zhang  
Yiqiang Zuo

# Table of Contents

Analysis of Bernstein's Factorization Circuit .....	1
<i>Arjen K. Lenstra, Adi Shamir, Jim Tomlinson, Eran Tromer</i>	
A Variant of the Cramer-Shoup Cryptosystem for Groups of Unknown Order .....	27
<i>Stefan Lucks</i>	
Looking beyond XTR .....	46
<i>Wieb Bosma, James Hutton, Eric R. Verheul</i>	
Bounds for Robust Metering Schemes and Their Relationship with $A^2$ -code .....	64
<i>Wakaha Ogata, Kaoru Kurosawa</i>	
Unconditionally Secure Anonymous Encryption and Group Authentication .....	81
<i>Goichiro Hanaoka, Junji Shikata, Yumiko Hanaoka, Hideki Imai</i>	
Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model .....	100
<i>Alexander W. Dent</i>	
On the Impossibilities of Basing One-Way Permutations on Central Cryptographic Primitives .....	110
<i>Yan-Cheng Chang, Chun-Yun Hsiao, Chi-Jen Lu</i>	
A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order .....	125
<i>Ivan Damgård, Eiichiro Fujisaki</i>	
Efficient Oblivious Transfer in the Bounded-Storage Model .....	143
<i>Dowon Hong, Ku-Young Chang, Heuisu Ryu</i>	
In How Many Ways Can You Write Rijndael? .....	160
<i>Elad Barkan, Eli Biham</i>	
On the Security of Rijndael-Like Structures against Differential and Linear Cryptanalysis .....	176
<i>Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong Yoon, Jongin Lim</i>	
Threshold Cryptosystems Based on Factoring .....	192
<i>Jonathan Katz, Moti Yung</i>	

Non-interactive Distributed-Verifier Proofs and Proving Relations among Commitments . . . . .	206
<i>Masayuki Abe, Ronald Cramer, Serge Fehr</i>	
Asynchronous Secure Communication Tolerating Mixed Adversaries . . . . .	224
<i>K. Srinathan, M.V.N. Ashwin Kumar, C. Pandu Rangan</i>	
Amplified Boomerang Attack against Reduced-Round SHACAL . . . . .	243
<i>Jongsung Kim, Dukjae Moon, Wonil Lee, Seokhie Hong, Sangjin Lee, Seokwon Jung</i>	
Enhancing Differential-Linear Cryptanalysis . . . . .	254
<i>Eli Biham, Orr Dunkelman, Nathan Keller</i>	
Cryptanalysis of Block Ciphers with Overdefined Systems of Equations . . .	267
<i>Nicolas T. Courtois, Josef Pieprzyk</i>	
Analysis of Neural Cryptography . . . . .	288
<i>Alexander Klimov, Anton Mityagin, Adi Shamir</i>	
The Hardness of Hensel Lifting: The Case of RSA and Discrete Logarithm . . . . .	299
<i>Dario Catalano, Phong Q. Nguyen, Jacques Stern</i>	
A Comparison and a Combination of SST and AGM Algorithms for Counting Points of Elliptic Curves in Characteristic 2 . . . . .	311
<i>Pierrick Gaudry</i>	
A General Formula of the $(t, n)$ -Threshold Visual Secret Sharing Scheme . .	328
<i>Hiroki Koga</i>	
On Unconditionally Secure Robust Distributed Key Distribution Centers . .	346
<i>Paolo D'Arco, Douglas R. Stinson</i>	
Short Signatures in the Random Oracle Model . . . . .	364
<i>Louis Granboulan</i>	
The Provable Security of Graph-Based One-Time Signatures and Extensions to Algebraic Signature Schemes . . . . .	379
<i>Alejandro Hevia, Daniele Micciancio</i>	
Transitive Signatures Based on Factoring and RSA . . . . .	397
<i>Mihir Bellare, Gregory Neven</i>	
1-out-of- $n$ Signatures from a Variety of Keys . . . . .	415
<i>Masayuki Abe, Miyako Ohkubo, Koutarou Suzuki</i>	
A Revocation Scheme with Minimal Storage at Receivers . . . . .	433
<i>Tomoyuki Asano</i>	

Optimistic Mixing for Exit-Polls .....	451
<i>Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, Ari Juels</i>	
Improved Construction of Nonlinear Resilient S-Boxes .....	466
<i>Kishan Chand Gupta, Palash Sarkar</i>	
An Upper Bound on the Number of $m$ -Resilient Boolean Functions .....	484
<i>Claude Carlet, Aline Gouget</i>	
Group Diffie-Hellman Key Exchange Secure Against Dictionary Attacks ..	497
<i>Emmanuel Bresson, Olivier Chevassut, David Pointcheval</i>	
Secure Channels Based on Authenticated Encryption Schemes: A Simple Characterization .....	515
<i>Chanathip Namprempre</i>	
ID-Based Blind Signature and Ring Signature from Pairings .....	533
<i>Fanguo Zhang, Kwangjo Kim</i>	
Hierarchical ID-Based Cryptography .....	548
<i>Craig Gentry, Alice Silverberg</i>	
Crypto-integrity .....	567
<i>Moti Yung</i>	
Gummy and Conductive Silicone Rubber Fingers .....	574
<i>Tsutomu Matsumoto</i>	
Author Index .....	577

# Analysis of Bernstein’s Factorization Circuit

Arjen K. Lenstra<sup>1</sup>, Adi Shamir<sup>2</sup>, Jim Tomlinson<sup>3</sup>, and Eran Tromer<sup>2</sup>

<sup>1</sup> Citibank, N.A. and Technische Universiteit Eindhoven,  
1 North Gate Road, Mendham, NJ 07945-3104, U.S.A.,  
[arjen.lenstra@citigroup.com](mailto:arjen.lenstra@citigroup.com)

<sup>2</sup> Department of Computer Science and Applied Mathematics,  
Weizmann Institute of Science, Rehovot 76100, Israel,  
{[shamir](mailto:shamir@wisdom.weizmann.ac.il), [tromer](mailto:tromer@wisdom.weizmann.ac.il)}@wisdom.weizmann.ac.il

<sup>3</sup> 99 E. 28th St., Bayonne, NJ 07002-4912, U.S.A., [jintom@optonline.net](mailto:jintom@optonline.net)

**Abstract.** In [1], Bernstein proposed a circuit-based implementation of the matrix step of the number field sieve factorization algorithm. These circuits offer an asymptotic cost reduction under the measure “construction cost  $\times$  run time”. We evaluate the cost of these circuits, in agreement with [1], but argue that compared to previously known methods these circuits can factor integers that are 1.17 times larger, rather than 3.01 as claimed (and even this, only under the non-standard cost measure). We also propose an improved circuit design based on a new mesh routing algorithm, and show that for factorization of 1024-bit integers the matrix step can, under an optimistic assumption about the matrix size, be completed within a day by a device that costs a few thousand dollars. We conclude that from a practical standpoint, the security of RSA relies exclusively on the hardness of the relation collection step of the number field sieve.

**Keywords:** factorization, number field sieve, RSA, mesh routing

## 1 Introduction

In [1], a new circuit-based approach is proposed for one of the steps of the number field sieve (NFS) integer factorization method, namely finding a linear relation in a large but sparse matrix. Unfortunately, the proposal from [1] has been misinterpreted on a large scale, even to the extent that announcements have been made that the results imply that common RSA key sizes no longer provide an adequate level of security.

In this paper we attempt to give a more balanced interpretation of [1]. In particular, we show that 1024-bit RSA keys are as secure as many believed them to be. Actually, [1] provides compelling new evidence that supports a traditional and widely used method to evaluate the security of RSA moduli. We present a variant of the analysis of [1] that would suggest that, under the metric proposed in [1], the number of digits of factorable integers  $n$  has grown by a factor  $1.17 + o(1)$ , for  $n \rightarrow \infty$  (in [1] a factor of  $3.01 + o(1)$  is mentioned).

We propose an improved circuit design, based on mesh routing rather than mesh sorting. To this end we describe a new routing algorithm whose performance in our setting seems optimal. With some further optimizations, the construction cost is reduced by several orders of magnitude compared to [1]. In the improved design the parallelization is gained essentially for free, since its cost is comparable to the cost of RAM needed just to store the input matrix.

We estimate the cost of breaking 1024-bit RSA with current technology. Using custom-built hardware to implement the improved circuit, the NFS matrix step becomes surprisingly inexpensive. However, the theoretical analysis shows that the cost of the relation collection step cannot be significantly reduced, regardless of the cost of the matrix step. We thus conclude that the practical security of RSA for commonly used modulus sizes is not significantly affected by [1].

Section 2 reviews background on the NFS; it does not contain any new material and simply serves as an explanation and confirmation of the analysis from [1]. Section 3 sketches the circuit approach of [1] and considers its implications. Section 4 discusses various cost-aspects of the NFS. Section 5 focuses on 1024-bit numbers, presenting custom hardware for the NFS matrix step both following [1] and using the newly proposed circuit. Section 6 summarizes our conclusions. Appendices A and B outline the limitations of off-the-shelf parts for the mesh-based approach and the traditional approach, respectively. Throughout this paper,  $n$  denotes the composite integer to be factored. Prices are in US dollars.

## 2 Background on the Number Field Sieve

In theory and in practice the two main steps of the NFS are the *relation collection step* and the *matrix step*. We review their heuristic asymptotic runtime analysis because it enables us to stress several points that are important for a proper understanding of “standard-NFS” and of “circuit-NFS” as proposed in [1].

**2.1 Smoothness.** An integer is called  $B$ -smooth if all its prime factors are at most  $B$ . Following [10, 3.16] we denote by  $L_x[r; \alpha]$  any function of  $x$  that equals

$$e^{(\alpha+o(1))(\log x)^r(\log \log x)^{1-r}}, \text{ for } x \rightarrow \infty,$$

where  $\alpha$  and  $r$  are real numbers with  $0 \leq r \leq 1$  and logarithms are natural. Thus,  $L_x[r; \alpha] + L_x[r; \beta] = L_x[r; \max(\alpha, \beta)]$ ,  $L_x[r; \alpha]L_x[r; \beta] = L_x[r; \alpha + \beta]$ ,  $L_x[r; \alpha]L_x[s; \beta] = L_x[r; \alpha]$  if  $r < s$ ,  $L_x[r; \alpha]^k = L_x[r; k\alpha]$  and if  $\alpha > 0$  then  $(\log x)^k L_x[r; \alpha] = L_x[r; \alpha]$  for any fixed  $k$ , and  $\pi(L_x[r; \alpha]) = L_x[r; \alpha]$  where  $\pi(y)$  is the number of primes  $\leq y$ .

Let  $\alpha > 0$ ,  $\beta > 0$ ,  $r$ , and  $s$  be fixed real numbers with  $0 < s < r \leq 1$ . A random positive integer  $\leq L_x[r; \alpha]$  is  $L_x[s; \beta]$ -smooth with probability

$$L_x[r - s; -\alpha(r - s)/\beta], \text{ for } x \rightarrow \infty.$$

We abbreviate  $L_n$  to  $L$  and  $L[1/3, \alpha]$  to  $L(\alpha)$ . Thus, a random integer  $\leq L[2/3, \alpha]$  is  $L(\beta)$ -smooth with probability  $L(-\alpha/(3\beta))$ . The notation  $L^{1.901 \dots + o(1)}$  in [1] corresponds to  $L(1.901 \dots)$  here. We write “ $\zeta \triangleq x$ ” for “ $\zeta = x + o(1)$  for  $n \rightarrow \infty$ .”

**2.2 Ordinary NFS.** To factor  $n$  using the NFS, more or less following the approach from [11], one selects a positive integer

$$d = \delta \left( \frac{\log n}{\log \log n} \right)^{1/3}$$

for a positive value  $\delta$  that is yet to be determined, an integer  $m$  close to  $n^{1/(d+1)}$ , a polynomial  $f(X) = \sum_{i=0}^d f_i X^i \in \mathbf{Z}[X]$  such that  $f(m) \equiv 0 \pmod n$  with each  $f_i$  of the same order of magnitude as  $m$ , a rational smoothness bound  $B_r$ , and an algebraic smoothness bound  $B_a$ . Other properties of these parameters are not relevant for our purposes.

A pair  $(a, b)$  of integers is called a *relation* if  $a$  and  $b$  are coprime,  $b > 0$ ,  $a - bm$  is  $B_r$ -smooth, and  $b^d f(a/b)$  is  $B_a$ -smooth. Each relation corresponds to a sparse  $D$ -dimensional bit vector with

$$D \approx \pi(B_r) + \#\{(p, r) : p \text{ prime} \leq B_a, f(r) \equiv 0 \pmod p\} \approx \pi(B_r) + \pi(B_a)$$

(cf. [11]). In the relation collection step a set of more than  $D$  relations is sought. Given this set, one or more linear dependencies modulo 2 among the corresponding  $D$ -dimensional bit vectors are constructed in the matrix step. Per dependency there is a chance of at least 50% (exactly 50% for RSA moduli) that a factor of  $n$  is found in the final step, the square root step. We discuss some issues of the relation collection and matrix steps that are relevant for [1].

**2.3 Relation Collection.** We restrict the search for relations to the rectangle  $|a| < L(\alpha)$ ,  $0 < b < L(\alpha)$  and use  $B_r$  and  $B_a$  that are both  $L(\beta)$  (which does not imply that  $B_r = B_a$ ), for  $\alpha, \beta > 0$  that are yet to be determined. It follows (cf. [2.1]) that  $D = L(\beta)$ . Furthermore,

$$|a - bm| = L[2/3, 1/\delta] \quad \text{and} \quad |b^d f(a/b)| = L[2/3, \alpha\delta + 1/\delta].$$

With [2.1] and under the usual assumption that  $a - bm$  and  $b^d f(a/b)$  behave, with respect to smoothness probabilities, independently as random integers of comparable sizes, the probability that both are  $L(\beta)$ -smooth is

$$L\left(\frac{-1/\delta}{3\beta}\right) \cdot L\left(\frac{-\alpha\delta - 1/\delta}{3\beta}\right) = L\left(-\frac{\alpha\delta + 2/\delta}{3\beta}\right).$$

The search space contains  $2L(\alpha)^2 = 2L(2\alpha) = L(2\alpha)$  pairs  $(a, b)$  and, due to the  $o(1)$ , as many pairs  $(a, b)$  with  $\gcd(a, b) = 1$ . It follows that  $\alpha$  and  $\beta$  must be chosen such that

$$L(2\alpha) \cdot L\left(-\frac{\alpha\delta + 2/\delta}{3\beta}\right) = L(\beta) (= D).$$

We find that

$$\alpha \stackrel{d}{=} \frac{3\beta^2 + 2/\delta}{6\beta - \delta}. \tag{1}$$

**2.4 Testing for Smoothness.** The  $(a, b)$  search space can be processed in  $L(2\alpha)$  operations. If sufficient memory is available this can be done using sieving. Current PC implementations intended for the factorization of relatively small numbers usually have adequate memory for sieving. For much larger numbers and current programs, sieving would become problematic. In that case, the search space can be processed in the “same”  $L(2\alpha)$  operations (with an, admittedly, larger  $o(1)$ ) but at a cost of only  $L(0)$  memory using the Elliptic Curve Method (ECM) embellished in any way one sees fit with trial division, Pollard rho, early aborts, etc., and run on any number  $K$  of processors in parallel to achieve a  $K$ -fold speedup. This was observed many times (see for instance [10, 4.15] and [4]). Thus, despite the fact that current implementations of the relation collection require substantial memory, it is well known that asymptotically this step requires negligible memory without incurring, in theory, a runtime penalty – in practice, however, it is substantially slower than sieving. Intermediate solutions that exchange sieving memory for many tightly coupled processors with small memories could prove valuable too; see [6] for an early example of this approach and [1] for various other interesting proposals that may turn out to be practically relevant. For the asymptotic argument, ECM suffices.

In improved NFS from [4] it was necessary to use a “memory-free” method when searching for  $B_\alpha$ -smooth numbers (cf. [2, 2]), in order to achieve the speedup. It was suggested in [4] that the ECM may be used for this purpose. Since memory usage was no concern for the analysis in [4], regular “memory-wasteful” sieving was suggested to test  $B_r$ -smoothness.

**2.5 The Matrix Step.** The choices made in [2, 3] result in a bit matrix  $A$  consisting of  $D = L(\beta)$  columns such that each column of  $A$  contains only  $L(0)$  nonzero entries. Denoting by  $w(A)$  the total number of nonzero entries of  $A$  (its *weight*), it follows that  $w(A) = L(\beta) \cdot L(0) = L(\beta)$ . Using a variety of techniques [5, 13], dependencies can be found after, essentially,  $O(D)$  multiplications of  $A$  times a bit vector. Since one matrix-by-vector multiplication can be done in  $O(w(A)) = L(\beta)$  operations, the matrix step can be completed in  $L(\beta)^2 = L(2\beta)$  operations. We use “standard-NFS” to refer to NFS that uses a matrix step with  $L(2\beta)$  operation count.

We will be concerned with a specific method for finding the dependencies in  $A$ , namely the block Wiedemann algorithm [5, 18] whose outline is as follows. Let  $K$  be the blocking factor, i.e., the amount of parallelism desired. We may assume that either  $K = 1$  or  $K > 32$ . Choose  $2K$  binary  $D$ -dimensional vectors  $\mathbf{v}_i, \mathbf{u}_j$  for  $1 \leq i, j \leq K$ . For each  $i$ , compute the vectors  $A^k \mathbf{v}_i$  for  $k$  up to roughly  $2D/K$ , using repeated matrix-by-vector multiplication. For each such vector  $A^k \mathbf{v}_i$ , compute the inner products  $\mathbf{u}_j A^k \mathbf{v}_i$ , for all  $j$ . Only these inner products are saved, to conserve storage. From the inner products, compute certain polynomials  $f_l(x)$ ,  $l = 1, \dots, K$  of degree about  $D/K$ . Then evaluate  $f_l(A) \mathbf{v}_i$ , for all  $l$  and  $i$  (take one  $\mathbf{v}_i$  at a time and evaluate  $f_l(A) \mathbf{v}_i$  for all  $l$  simultaneously using repeated matrix-by-vector multiplications). From the result,  $K$  elements from the kernel of  $A$  can be computed. The procedure is probabilistic, but succeeds with high probability for  $K \gg 1$  [17]. For  $K = 1$ , the cost roughly doubles [18].



For reasonable blocking factors ( $K = 1$  or  $32 \leq K \ll \sqrt{D}$ ), the block Wiedemann algorithm involves about  $3D$  matrix-by-vector multiplications. These multiplications dominate the cost of the matrix step; accordingly, the circuits of [1], and our variants thereof, aim to reduce their cost. Note that the multiplications are performed in  $2K$  separate chains where each chain involves repeated left-multiplication by  $A$ . The proposed circuits rely on this for their efficiency. Thus, they appear less suitable for other dependency-finding algorithms, such as block Lanczos [13] which requires just  $2D$  multiplications.

**2.6 NFS Parameter Optimization for Matrix Exponent  $2\epsilon > 1$ .** With the relation collection and matrix steps in  $L(2\alpha)$  and  $L(2\beta)$  operations, respectively, the values for  $\alpha$ ,  $\beta$ , and  $\delta$  that minimize the overall NFS operation count follow using Relation (1). However, we also need the optimal values if the “cost” of the matrix step is different from  $L(\beta)^2$ : in [1] “cost” is defined using a metric that is not always the same as operation count, so we need to analyse the NFS using alternative cost metrics. This can be done by allowing flexibility in the “cost” of the matrix step: we consider how to optimize the NFS parameters for an  $L(\beta)^{2\epsilon}$  matrix step, for some exponent  $\epsilon > 1/2$ . The corresponding relation collection operation count is fixed at  $L(2\alpha)$  (cf. 2.4).

We balance the cost of the relation collection and matrix steps by taking  $\alpha \triangleq \epsilon\beta$ . With (1) it follows that

$$3(2\epsilon - 1)\beta^2 - \epsilon\beta\delta - 2/\delta \triangleq 0, \text{ so that } \beta \triangleq \frac{\epsilon\delta + \sqrt{\epsilon^2\delta^2 + 24(2\epsilon - 1)/\delta}}{6(2\epsilon - 1)}.$$

Minimizing  $\beta$  given  $\epsilon$  leads to

$$\delta \triangleq \sqrt[3]{3(2\epsilon - 1)/\epsilon^2} \quad (2)$$

and

$$\beta \triangleq 2\sqrt[3]{\epsilon/(3(2\epsilon - 1))^2}. \quad (3)$$

Minimizing the resulting

$$\alpha \triangleq 2\epsilon\sqrt[3]{\epsilon/(3(2\epsilon - 1))^2} \quad (4)$$

leads to  $\epsilon = 1$  and  $\alpha \triangleq 2/3^{2/3}$ : even though  $\epsilon < 1$  would allow more “relaxed” relations (i.e., larger smoothness bounds and thus easier to find), the fact that more of such relations have to be found becomes counterproductive. It follows that an operation count of  $L(4/3^{2/3})$  is optimal for relation collection, but that for  $2\epsilon > 2$  it is better to use suboptimal relation collection because otherwise the matrix step would dominate. We find the following optimal NFS parameters:

$1 < 2\epsilon \leq 2$ :

$\delta \triangleq 3^{1/3}$ ,  $\alpha \triangleq 2/3^{2/3}$ , and  $\beta \triangleq 2/3^{2/3}$ , with operation counts of relation collection and matrix steps equal to  $L(4/3^{2/3})$  and  $L(4\epsilon/3^{2/3})$ , respectively. For  $\epsilon = 1$  the operation counts of the two steps are the same (when expressed in  $L$ ) and the overall operation count is  $L(4/3^{2/3}) = L((64/9)^{1/3}) =$

$L(1.9229994 \dots)$ . This corresponds to the heuristic asymptotic runtime of the NFS as given in [11]. We refer to these parameter choices as the *ordinary parameter choices*.

$2\epsilon > 2$ :

$\delta, \alpha$ , and  $\beta$  as given by Relations (2), (4), and (3), respectively, with operation count  $L(2\alpha)$  for relation collection and cost  $L(2\epsilon\beta)$  for the matrix step, where  $L(2\alpha) = L(2\epsilon\beta)$ . More in particular, we find the following values.

$2\epsilon = 5/2$ :

$\delta \triangleq (5/3)^{1/3}(6/5)$ ,  $\alpha \triangleq (5/3)^{1/3}(5/6)$ , and  $\beta \triangleq (5/3)^{1/3}(2/3)$ , for an operation count and cost  $L((5/3)^{4/3}) = L(1.9760518 \dots)$  for the relation collection and matrix steps, respectively. These values are familiar from [1, Section 6: Circuits]. With  $(1.9229994 \dots / 1.9760518 \dots + o(1))^3 \triangleq 0.9216$  and equating operation count and cost, this suggests that factoring  $0.9216 \cdot 512 \approx 472$ -bit composites using NFS with matrix exponent  $5/2$  is comparable to factoring 512-bit ones using standard-NFS with ordinary parameter choices (disregarding the effects of the  $o(1)$ 's).

$2\epsilon = 3$ :

$\delta \triangleq 2/3^{1/3}$ ,  $\alpha \triangleq 3^{2/3}/2$ , and  $\beta \triangleq 3^{-1/3}$ , for an operation count and cost of  $L(3^{2/3}) = L(2.0800838 \dots)$  for the relation collection and matrix steps, respectively.

**2.7 Improved NFS.** It was shown in [4] that ordinary NFS from [11], and as used in [2.2], can be improved by using more than a single polynomial  $f$ . Let  $\alpha$  and  $\delta$  be as in [2.3] and [2.2] respectively, let  $\beta$  indicate the rational smoothness bound  $B_r$  (i.e.,  $B_r = L(\beta)$ ), and let  $\gamma$  indicate the algebraic smoothness bound  $B_a$  (i.e.,  $B_a = L(\gamma)$ ). Let  $G$  be a set of  $B_r/B_a = L(\beta - \gamma)$  different polynomials, each of degree  $d$  and common root  $m$  modulo  $n$  (as in [2.2]). A pair  $(a, b)$  of integers is a relation if  $a$  and  $b$  are coprime,  $b > 0$ ,  $a - bm$  is  $B_r$ -smooth, and  $b^d g(a/b)$  is  $B_a$ -smooth for at least one  $g \in G$ . Let  $\epsilon$  be the matrix exponent. Balancing the cost of the relation collection and matrix steps it follows that  $\alpha \triangleq \epsilon\beta$ .

Optimization leads to

$$\gamma \triangleq \left( \frac{\epsilon^2 + 5\epsilon + 2 + (\epsilon + 1)\sqrt{\epsilon^2 + 8\epsilon + 4}}{9(2\epsilon + 1)} \right)^{1/3}$$

and for this  $\gamma$  to

$$\alpha \triangleq \frac{9\gamma^3 + 1 + \sqrt{18\gamma^3(2\epsilon + 1) + 1}}{18\gamma^2}, \quad \beta \triangleq \alpha/\epsilon,$$

and

$$\delta \triangleq \frac{3\gamma(-4\epsilon - 1 + \sqrt{18\gamma^3(2\epsilon + 1) + 1})}{9\gamma^3 - 4\epsilon}.$$

It follows that for  $2\epsilon = 2$  the method from [4] gives an improvement over the ordinary method, namely  $L(1.9018836 \dots)$ . The condition  $\beta \geq \gamma$  leads to  $2\epsilon \leq$

$7/3$ , so that for  $2\epsilon > 7/3$  (as in circuit-NFS, cf. [3.1]) usage of the method from [4] no longer leads to an improvement over the ordinary method. This explains why in [1] the method from [4] is used to select parameters for standard-NFS and why the ordinary method is used for circuit-NFS.

With [2.1] it follows that the sum of the (rational) sieving and ECM-based (algebraic) smoothness times from [4] (cf. last paragraph of [2.4]) is minimized if  $\beta = \gamma + 1/(3\beta\delta)$ . The above formulas then lead to  $2\epsilon = (3 + \sqrt{17})/4 = 1.7807764\dots$ . Therefore, unlike the ordinary parameter selection method, optimal relation collection for the improved method from [4] occurs for an  $\epsilon$  with  $2\epsilon < 2$ : with  $\epsilon = 0.8903882\dots$  the operation count for relation collection becomes  $L(1.8689328\dots)$ . Thus, in principle, and depending on the cost function one is using, the improved method would be able to take advantage of a matrix step with exponent  $2\epsilon < 2$ . If we disregard the matrix step and minimize the operation count of relation collection, this method yields a cost of  $L(1.8689328\dots)$ .

### 3 The Circuits for Integer Factorization from [1]

**3.1 Matrix-by-Vector Multiplication Using Mesh Sorting.** In [1] an interesting new mesh-sorting-based method is described to compute a matrix-by-vector product. Let  $A$  be the bit matrix from [2.5] with  $D = L(\beta)$  columns and weight  $w(A) = L(\beta)$ , and let  $m$  be the least power of 2 such that  $m^2 > w(A) + 2D$ . Thus  $m = L(\beta/2)$ . We assume, without loss of generality, that  $A$  is square. A mesh of  $m \times m$  processors, each with  $O(\log D) = L(0)$  memory, initially stores the matrix  $A$  and a not necessarily sparse  $D$ -dimensional bit vector  $\mathbf{v}$ . An elegant method is given that computes the product  $A\mathbf{v}$  using repeated sorting in  $O(m)$  steps, where each step involves a small constant number of simultaneous operations on all  $m \times m$  mesh processors. At the end of the computation  $A\mathbf{v}$  can easily be extracted from the mesh. Furthermore, the mesh is immediately, without further changes to its state, ready for the computation of the product of  $A$  and the vector  $A\mathbf{v}$ . We use “circuit-NFS” to refer to NFS that uses the mesh-sorting-based matrix step.

**3.2 The Throughput Cost Function from [1].** Judging by operation counts, the mesh-based algorithm is not competitive with the traditional way of computing  $A\mathbf{v}$ : as indicated in [2.5] it can be done in  $O(w(A)) = L(\beta)$  operations. The mesh-based computation takes  $O(m)$  steps on all  $m \times m$  mesh processors simultaneously, resulting in an operation count per matrix-by-vector multiplication of  $O(m^3) = L(3\beta/2)$ . Iterating the matrix-by-vector multiplications  $L(\beta)$  times results in a mesh-sorting-based matrix step that requires  $L(5\beta/2) = L(\beta)^{5/2}$  operations as opposed to just  $L(2\beta)$  for the traditional approach. This explains the non-ordinary relation collection parameters used in [1] corresponding to the analysis given in [2.6] for  $2\epsilon = 5/2$ , something we comment upon below in [3.3].

However, the standard comparison of operation counts overlooks the following fact. The traditional approach requires memory  $O(w(A) + D) = L(\beta)$  for storage of  $A$  and the vector; given that amount of memory it takes time  $L(2\beta)$ .

But given the  $m \times m$  mesh, with  $m \times m = L(\beta)$ , the mesh-based approach takes time just  $L(3\beta/2)$  because during each unit of time  $L(\beta)$  operations are carried out simultaneously on the mesh. To capture the advantage of “active small processors” (as in the mesh) compared to “inactive memory” (as in the traditional approach) and the fact that their price is comparable, it is stipulated in [1] that the cost of factorization is “the product of the time and the cost of the machine.” We refer to this cost function as **throughput cost**, since it can be interpreted as measuring the equipment cost per unit problem-solving throughput. It is frequently used in VLSI design (where it’s known as “AT cost”, for Area $\times$ Time), but apparently was not used previously in the context of computational number theory.

It appears that throughput cost is indeed appropriate when a large number of problems must be solved during some long period of time while minimizing total expenses. This does not imply that throughput cost is always appropriate for assessing security, as illustrated by the following example. Suppose Carol wishes to assess the risk of her encryption key being broken by each of two adversaries, Alice and Bob. Carol knows that Alice has plans for a device that costs \$1M and takes 50 years to break a key, and that Bob’s device costs \$50M and takes 1 year to break a key. In one scenario, each adversary has a \$1M budget — clearly Alice is dangerous and Bob is not. In another scenario, each adversary has a \$50M budget. This time both are dangerous, but Bob apparently forms a greater menace because he can break Carol’s key within one year, while Alice still needs 50 years. Thus, the two devices have the same throughput cost, yet either can be more “dangerous” than the other, depending on external settings. The key point is that if Alice and Bob have many keys to break within 50 years then indeed their cost-per-key figures are identical, but the time it will take Bob to break Carol’s key depends on her priority in his list of victims, and arguably Carol should make the paranoid assumption that she is first.

In Section 4 we comment further on performance measurement for the NFS.

**3.3 Application of the Throughput Cost.** The time required for all matrix-by-vector multiplications on the mesh is  $L(3\beta/2)$ . The equipment cost of the mesh is the cost of  $m^2$  small processors with  $L(0)$  memory per processor, and is thus  $L(\beta)$ . The throughput cost, the product of the time and the cost of the equipment, is therefore  $L(5\beta/2)$ . The matrix step of standard-NFS requires time  $L(2\beta)$  and equipment cost  $L(\beta)$  for the memory, resulting in a throughput cost of  $L(3\beta)$ . Thus, the throughput cost advantage of the mesh-based approach is a factor  $L(\beta/2)$  if the two methods would use the same  $\beta$  (cf. Remark 3.4).

The same observation applies if the standard-NFS matrix step is  $K$ -fold parallelized, for reasonable  $K$  (cf. 2.5): the time drops by a factor  $K$  which is cancelled (in the throughput cost) by a  $K$  times higher equipment cost because each participating processor needs the same memory  $L(\beta)$ . In circuit-NFS (i.e., the mesh) a parallelization factor  $m^2$  is used: the time drops by a factor only  $m$  (not  $m^2$ ), but the equipment cost stays the same because memory  $L(0)$  suffices for each of the  $m^2$  participating processors. Thus, with the throughput cost circuit-NFS achieves an advantage of  $m = L(\beta/2)$ . The mesh itself can of course

be  $K$ -fold parallelized but the resulting  $K$ -fold increase in equipment cost and  $K$ -fold drop in time cancel each other in the throughput cost [1] Section 4].

**Remark 3.4.** It can be argued that before evaluating an existing algorithm based on a new cost function, the algorithm first should be tuned to the new cost function. This is further commented upon below in 3.5.

**3.5 Implication of the Throughput Cost.** We consider the implication of the matrix step throughput cost of  $L(5\beta/2)$  for circuit-NFS compared to  $L(3\beta)$  for standard-NFS. In [1] the well known fact is used that the throughput cost of relation collection is  $L(2\alpha)$  (cf. 2.4): an operation count of  $L(2\alpha)$  on a single processor with  $L(0)$  memory results in time  $L(2\alpha)$ , equipment cost  $L(0)$ , and throughput cost  $L(2\alpha)$ . This can be time-sliced in any way that is convenient, i.e., for any  $K$  use  $K$  processors of  $L(0)$  memory each and spend time  $L(2\alpha)/K$  on all  $K$  processors simultaneously, resulting in the same throughput cost  $L(2\alpha)$ . Thus, for relation collection the throughput cost is proportional to the operation count. The analysis of 2.6 applies with  $2\epsilon = 5/2$  and leads to an optimal overall circuit-NFS throughput cost of  $L(1.9760518\cdots)$ . As mentioned above and in 3.2, the throughput cost and the operation count are equivalent for both relation collection and the matrix step of circuit-NFS. Thus, as calculated in 2.6, circuit-NFS is from an operation count point of view less powerful than standard-NFS, losing already 40 bits in the 500-bit range (disregarding the  $o(1)$ 's) when compared to standard-NFS with ordinary parameter choices. This conclusion applies to any NFS implementation, such as many existing ones, where memory requirements are not multiplicatively included in the cost function.

But operation count is not the point of view taken in [1]. There standard-NFS is compared to circuit-NFS in the following way. The parameters for standard-NFS are chosen under the assumption that the throughput cost of relation collection is  $L(3\alpha)$ : operation count  $L(2\alpha)$  and memory cost  $L(\alpha)$  for the sieving result in time  $L(2\alpha)/K$  and equipment cost  $K \cdot L(\alpha)$  (for any  $K$ -fold parallelization) and thus throughput cost  $L(3\alpha)$ . This disregards the fact that long before [1] appeared it was known that the use of  $L(\alpha)$  memory per processor may be convenient, in practice and for relatively small numbers, but is by no means required (cf. 2.4). In any case, combined with  $L(3\beta)$  for the throughput cost of the matrix step this leads to  $\alpha \triangleq \beta$ , implying that the analysis from 2.6 with  $2\epsilon = 2$  applies, but that the resulting operation count must be raised to the  $3/2$ -th power. In [1] the improvement from [4] mentioned in 2.7 is used, leading to a throughput cost for standard-NFS of  $L(2.8528254\cdots)$  (where  $2.8528254\cdots$  is 1.5 times the  $1.9018836\cdots$  referred to in 2.7). Since  $(2.8528254\cdots/1.9760518\cdots)^3 = 3.0090581\cdots$ , it is suggested in [1] that the number of digits of factorable composites grows by a factor 3 if circuit-NFS is used instead of standard-NFS.

**3.6 Alternative Interpretation.** How does the comparison between circuit-NFS and standard-NFS with respect to their throughput costs turn out if standard-NFS is first properly tuned (Remark 3.4) to the throughput cost function,

given the state of the art in, say, 1990 (cf. [10] 4.15]; also the year that [4] originally appeared)? With throughput cost  $L(2\alpha)$  for relation collection (cf. above and [2.4]), the analysis from [2.6] with  $2\epsilon = 3$  applies, resulting in a throughput cost of just  $L(2.0800838 \dots)$  for standard-NFS. Since  $(2.0800838 \dots / 1.9760518 \dots)^3 < 1.17$ , this would suggest that  $1.17D$ -digit composites can be factored using circuit-NFS for the throughput cost of  $D$ -digit integers using standard-NFS. The significance of this comparison depends on whether or not the throughput cost is an acceptable way of measuring the cost of standard-NFS. If not, then the conclusion based on the operation count (as mentioned above) would be that circuit-NFS is slower than standard-NFS; but see Section 4 for a more complete picture. Other examples where it is recognized that the memory cost of relation collection is asymptotically not a concern can be found in [12] and [9], and are implied by [14].

**Remark 3.7.** It can be argued that the approach in [3.6] of replacing the ordinary standard-NFS parameters by smaller smoothness bounds in order to make the matrix step easier corresponds to what happens in many actual NFS factorizations. There it is done not only to make the matrix step less cumbersome at the cost of somewhat more sieving, but also to make do with available PC memories. Each contributing PC uses the largest smoothness bounds and sieving range that fit conveniently and that cause minimal interference with the PC-owner’s real work. Thus, parameters may vary from machine to machine. This is combined with other memory saving methods such as “special- $q$ ’s.” In any case, if insufficient memory is available for sieving with optimal ordinary parameters, one does not run out to buy more memory but settles for slight suboptimality, with the added benefit of an easier matrix step. See also [4.1].

**Remark 3.8.** In [19], Wiener outlines a three-dimensional circuit for the matrix step, with structure that is optimal in a certain sense (when considering the cost of internal wiring). This design leads to a matrix step exponent of  $2\epsilon = 7/3$ , compared to  $5/2$  in the designs of [1] and this paper. However, adaptation of that design to two dimensions yields a matrix step exponent that is asymptotically identical to ours, and vice versa. Thus the approach of [19] is asymptotically equivalent to ours, while its practical cost remains to be evaluated. We note that in either approach, there are sound technological reasons to prefer the 2D variant. Interestingly,  $2\epsilon = 7/3$  is the point where improved and standard NFS become the same (cf. [2.7]).

## 4 Operation Count, Equipment Cost, and Real Time

The asymptotic characteristics of standard-NFS and circuit-NFS with respect to their operation count, equipment, and real time spent are summarized in Table 1. For non- $L(0)$  equipment requirements it is specified if the main cost goes to memory (“RAM”), processing elements (“PEs”) with  $L(0)$  memory, or a square mesh as in [3.1], and “tuned” refers to the alternative analysis in [3.6].

**Table 1.** NFS costs: operation count, equipment, and real time.

	overall operation count	relation collection		matrix step	
		equipment	real time	equipment	real time
standard-NFS: $\begin{cases} \text{sieving} \\ \text{no sieving} \end{cases}$	$L(1.90)$	$\begin{cases} L(0.95) \text{ RAM} \\ L(0) \end{cases}$	$L(1.90)$	$L(0.95) \text{ RAM}$	$L(1.90)$
tuned no sieving	<u><math>L(2.08)</math></u>	sequential: $L(0)$ parallel: $L(0.69) \text{ PEs}$	$L(2.08)$ $L(1.39)$	$L(0.69) \text{ RAM}$	$L(1.39)$
circuit-NFS:	<u><math>L(1.98)</math></u>	sequential: $L(0)$ parallel: $L(0.79) \text{ PEs}$	$L(1.98)$ $L(1.19)$	$L(0.79) \text{ mesh}$	$L(1.19)$

The underlined operation counts are the same as the corresponding throughput costs. For the other operation count the throughput cost (not optimized if no sieving is used) follows by taking the maximum of the products of the figures in the “equipment” and “real time” columns. Relation collection, whether using sieving or not, allows almost arbitrary parallelization (as used in the last two rows of Table 1). The amount of parallelization allowed in the matrix step of standard-NFS is much more limited (cf. 2.5); it is not used in Table 1.

**4.1 Lowering the Cost of the Standard-NFS Matrix Step.** We show at what cost the asymptotic advantages of the circuit-NFS matrix step (low throughput cost and short real time) can be matched, asymptotically, using the traditional approach to the matrix step. This requires a smaller matrix, i.e., lower smoothness bounds, and results therefore in slower relation collection. We illustrate this with two examples. To get matching throughput costs for the matrix steps of standard-NFS and circuit-NFS,  $\beta$  must be chosen such that  $L(3\beta) = L((5/3)^{4/3}) = L(1.9760\dots)$ , so that the matrix step of standard-NFS requires  $L(\beta) = L(0.6586\dots)$  RAM and real time  $L(2\beta) = L(1.3173\dots)$ . Substituting this  $\beta$  in Relation 1 and minimizing  $\alpha$  with respect to  $\delta$  we find

$$\delta \triangleq \frac{\sqrt{4 + 36\beta^3} - 2}{3\beta^2}, \quad (5)$$

i.e.,  $\delta \triangleq 1.3675\dots$  and  $\alpha \triangleq 1.0694\dots$ , resulting in relation collection operation count  $L(2.1389\dots)$ . Or, one could match the real time of the matrix steps: with  $L(2\beta) = L((5/3)^{1/3}) = L(1.1856\dots)$  the matrix step of standard-NFS requires  $L(0.5928\dots)$  RAM and real time  $L(1.1856\dots)$ . With Relation 5 we find that  $\delta \triangleq 1.3195\dots$ ,  $\alpha \triangleq 1.1486\dots$ , and relation collection operation count  $L(2.2973\dots)$ .

**4.2 Operation Count Based Estimates.** Operation count is the traditional way of measuring the cost of the NFS. It corresponds to the standard complexity

measure of “runtime” and neglects the cost of memory or other equipment that is needed to actually “run” the algorithm. It was used, for instance, in [11] and [4] and was analysed in [2.6] and [2.7].

It can be seen in Table 1 and was indicated in [3.5] that the operation count for circuit-NFS is higher than for standard-NFS (assuming both methods are optimized with respect to the operation count):  $L(1.9760518\dots)$  as opposed to just  $L(1.9018836\dots)$  when using the improved version (cf. [2.7]) as in Table 1, or as opposed to  $L(1.9229994\dots)$  when using the ordinary version (cf. [2.6]) as in [3.5]. Thus, RSA moduli that are deemed sufficiently secure based on standard-NFS operation count security estimates, are even more secure when circuit-NFS is considered instead. Such estimates are common; see for instance [14] and the “computationally equivalent” estimates in [9.12]. Security estimates based on the recommendations from [14] or the main ones (i.e., the conservative “computationally equivalent” ones) from [9.12] are therefore not affected by the result from [1]. Nevertheless, we agree with [2] that the PC-based realization suggested in [12], meant to present an at the time possibly realistic approach that users can relate to, may not be the best way to realize a certain operation count; see also the last paragraph of [12, 2.4.7]. The estimates from [15] are affected by [1].

**Remark 4.3.** Historically, in past factorization experiments the matrix step was always solved using a fraction of the effort required by relation collection. Moreover, the memory requirements of sieving-based relation collection have never turned out to be a serious problem (it was not even necessary to fall back to the memory-efficient ECM and its variations). Thus, despite the asymptotic analysis, extrapolation from past experience would predict that the bottleneck of the NFS method is relation collection, and that simple operation count is a better practical cost measure for NFS than other measures that are presumably more realistic. The choice of cost function in [9.12] was done accordingly.

The findings of [1] further support this conservative approach, by going a long way towards closing the gap between the two measures of cost when applied to the NFS: 93% of the gap according to [3.5], and 61% according to [3.6].

## 5 Hardware for the Matrix Step for 1024-Bit Moduli

In this section we extrapolate current factoring knowledge to come up with reasonable estimates for the sizes of the matrix  $A$  that would have to be processed for the factorization of a 1024-bit composite when using ordinary relation collection (cf. [2.6]), and using slower relation collection according to matrix exponent  $5/2$  as used in circuit-NFS. For the latter (smaller sized) matrix we consider how expensive it would be to build the mesh-sorting-based matrix-by-vector multiplication circuit proposed in [1] using custom-built hardware and we estimate how much time the matrix step would take on the resulting device. We then propose an alternative mesh-based matrix-by-vector multiplication circuit and estimate its performance for both matrices, for custom-built and off-the-shelf hardware.



Throughout this section we are interested mainly in assessing feasibility, for the purpose of evaluating the security implications. Our assumptions will be somewhat optimistic, but we believe that the designs are fundamentally sound and give realistic indications of feasibility using technology that is available in the present or in the near future.

**5.1 Matrix Sizes.** For the factorization of RSA-512 the matrix had about 6.7 million columns and average column density about 63 [3]. There is no doubt that this matrix is considerably smaller than a matrix that would have resulted from ordinary relation collection as defined in 2.6, cf. Remark 3.7. Nevertheless, we make the optimistic assumption that this is the size that would result from ordinary relation collection.

Combining this figure with the  $L(2/3^{2/3})$  matrix size growth rate (cf. 2.6) we find

$$6\,700\,000 \cdot \frac{L_{2^{1024}}[1/3, 2/3^{2/3}]}{L_{2^{512}}[1/3, 2/3^{2/3}]} \approx 1.8 \cdot 10^{10}$$

(cf. 2.1). Including the effect of the  $o(1)$  it is estimated that an optimal 1024-bit matrix would contain about  $10^{10}$  columns. We optimistically assume an average column density of about 100. We refer to this matrix as the “large” matrix.

Correcting this matrix size for the  $L((5/3)^{1/3}(2/3))$  matrix size growth rate for matrix exponent  $5/2$  (cf. 2.6) we find

$$1.8 \cdot 10^{10} \cdot \frac{L_{2^{1024}}[1/3, (5/3)^{1/3}(2/3)]}{L_{2^{1024}}[1/3, 2/3^{2/3}]} \approx 8.7 \cdot 10^7.$$

We arrive at an estimate of about  $4 \cdot 10^7$  columns for the circuit-NFS 1024-bit matrix. We again, optimistically, assume that the average column density is about 100. We refer to this matrix as the “small” matrix.

**5.2 Estimated Relation Collection Cost.** Relation collection for RSA-512 could have been done in about 8 years on a 1GHz PC [3]. Since

$$8 \cdot \frac{L_{2^{1024}}[1/3, 4/3^{2/3}]}{L_{2^{512}}[1/3, 4/3^{2/3}]} \approx 6 \cdot 10^7$$

we estimate that generating the large matrix would require about a year on about 30 million 1GHz PCs with large memories (or more PC-time but less memory when using alternative smoothness tests – keep in mind, though, that it may be possible to achieve the same operation count using different hardware, as rightly noted in [1] and speculated in [12, 2.4.7]). With

$$\frac{L_{2^{1024}}[1/3, (5/3)^{4/3}]}{L_{2^{1024}}[1/3, 4/3^{2/3}]} \approx 5$$

it follows that generating the smaller matrix would require about 5 times the above effort. Neither computation is infeasible. But, it can be argued that 1024-bit RSA moduli provide a reasonable level of security just based on the operation count of the relation collection step.

**5.3 Processing the “Small” Matrix Using Bernstein’s Circuits.** We estimate the size of the circuit required to implement the mesh circuit of [1] when the NFS parameters are optimized for the throughput cost function and 1024-bit composites. We then derive a rough prediction of the associated costs when the mesh is implemented by custom hardware using current VLSI technology. In this subsection we use the circuit exactly as described in [1]; the next subsections will make several improvements, including those listed as future plans in [1].

In [1], the algorithm used for finding dependencies among the columns of  $A$  is Wiedemann’s original algorithm [18], which is a special case of block Wiedemann with blocking factor  $K=1$  (cf. [2.5]). In the first stage (inner product computation), we are given the sparse  $D \times D$  matrix  $A$  and some pair of vectors  $\mathbf{u}, \mathbf{v}$  and wish to calculate  $\mathbf{u}A^k\mathbf{v}$  for  $k = 1, \dots, 2D$ . The polynomial evaluation stage is slightly different, but the designs given below can be easily adapted so we will not discuss it explicitly.

The mesh consists of  $m \times m$  nodes, where  $m^2 > w(A) + 2D$  (cf. [3.1]). By assumption,  $w(A) \approx 4 \cdot 10^9$  and  $D \approx 4 \cdot 10^7$  so we may choose  $m = 63256$ . To execute the sorting-based algorithm, each node consists mainly of 3 registers of  $\lceil \log_2(4 \cdot 10^7) \rceil = 26$  bits each, a 26-bit compare-exchange element (in at least half of the nodes), and some logic for tracking the current stage of the algorithm. Input, namely the nonzero elements of  $A$  and the initial vector  $\mathbf{v}$ , is loaded just once so this can be done serially. The mesh computes the vectors  $A^k\mathbf{v}$  by repeated matrix-by-vector multiplication, and following each such multiplication it calculates the inner product  $\mathbf{u}(A^k\mathbf{v})$  and outputs this single bit.

In standard CMOS VLSI design, a single-bit register (i.e., a D-type edge-triggered flip-flop) requires about 8 transistors, which amounts to 624 transistors per node. To account for the logic and additional overheads such as a clock distribution network, we shall assume an average of 2000 transistors per node for a total of  $8.0 \cdot 10^{12}$  transistors in the mesh.

As a representative of current technology available on large scale we consider Intel’s latest Pentium processor, the Pentium 4 “Northwood” ( $0.13\mu\text{m}^2$  feature size process). A single Northwood chip (inclusive of its on-board L2 cache) contains  $5.5 \cdot 10^7$  transistors, and can be manufactured in dies of size  $131\text{mm}^2$  on wafers of diameter 300mm, i.e., about 530 chips per wafer when disregarding defects. The 1.6GHz variant is currently sold at \$140 in retail channels. By transistor count, the complete mesh would require about  $(8.0 \cdot 10^{12}) / (5.5 \cdot 10^7) \approx 145\,500$  Northwood-sized dies or about 273 wafers. Using the above per-chip price figure naively, the construction cost is about \$20M. Alternatively, assuming a wafer cost of about \$5,000 we get a construction cost of roughly \$1.4M, and the initial costs (e.g., mask creation) are under \$1M.

The matter of inter-chip communication is problematic. The mesh as a whole needs very few external lines (serial input, 1-bit output, clock, and power). However, a chip consisting of  $s \times s$  nodes has  $4s - 4$  nodes on its edges, and each of these needs two 26-bit bidirectional links with its neighbor on an adjacent chip, for a total of about  $2 \cdot 2 \cdot 26 \cdot 4s = 416s$  connections. Moreover, such connections typically do not support the full 1GHz clock rate, so to achieve the

necessary bandwidth we will need about 4 times as many connections: 1664s. While standard wiring technology cannot provide such enormous density, the following scheme seems plausible. Emerging “flip-chip” technologies allow direct connections between chips that are placed face-to-face, at a density of 277 connections per  $\text{mm}^2$  (i.e.,  $60\mu\text{s}$  array pitch). We cut each wafer into the shape of a cross, and arrange the wafers in a two-dimensional grid with the arms of adjacent wafers in full overlap. The central square of each cross-shaped wafer contains mesh nodes, and the arms are dedicated to inter-wafer connections. Simple calculation shows that with the above connection density, if 40% of the (uncut) wafer area is used for mesh nodes then there is sufficient room left for the connection pads and associated circuitry. This disregards the issues of delays (mesh edges that cross wafer boundaries are realized by longer wires and are thus slower than the rest), and of the defects which are bound to occur. To address these, adaptation of the algorithm is needed. Assuming the algorithmic issues are surmountable, the inter-wafer communication entails a cost increase by a factor of about 3, to \$4.1M.

According to [II Section 4], a matrix-by-vector multiplication consists of, essentially, three sort operations on the  $m \times m$  mesh. Each sort operation takes  $8m$  steps, where each step consists of a compare-exchange operation between 26-bit registers of adjacent nodes. Thus, multiplication requires  $3 \cdot 8m \approx 1.52 \cdot 10^6$  steps. Assuming that each step takes a single clock cycle at a 1GHz clock rate, we get a throughput of 659 multiplications per second.

Basically, Wiedemann's algorithm requires  $3D$  multiplications. Alas, the use of blocking factor  $K = 1$  entails some additional costs. First, the number of multiplications roughly doubles due to the possibility of failure (cf. 2.5). Moreover, the algorithm will yield a single vector from the kernel of  $A$ , whereas the Number Field Sieve requires several linearly independent kernel elements: half of these yield a trivial congruence (c.f. 2.2), and moreover certain NFS optimizations necessitate discarding most of the vectors. In RSA-512, a total of about 10 kernel vectors were needed. Fortunately, getting additional vectors is likely to be cheaper than getting the first one (this is implicit in [18, Algorithm 1]). Overall, we expect the number of multiplications to be roughly  $2 \cdot \frac{10}{3} \cdot 3D = 20D$ . Thus, the expected total running time is roughly  $20 \cdot 4 \cdot 10^7 / 659 \approx 1\,210\,000$  seconds, or 14 days. The throughput cost is thus  $5.10 \cdot 10^{12} \$ \times \text{sec}$ .

If we increase the blocking factor from 1 to over 32 and handle the multiplication chains sequentially on a single mesh, then only  $3D$  multiplications are needed ([II considers this but claims that it will not change the cost of computation; that is true only up to constant factors). In this case the time decreases to 50 hours, and the throughput cost decreases to  $7.4 \cdot 10^{11} \$ \times \text{sec}$ .

Heat dissipation (i.e., power consumption) may limit the node density and clock rate of the device, and needs to be analysed. Note however that this limitation is technological rather than theoretical, since in principle the mesh sorting algorithm can be efficiently implemented using reversible gates and arbitrarily low heat dissipation.

**5.4 A Routing-Based Circuit.** The above analysis refers to the mesh circuit described in [1], which relies on the novel use of mesh sorting for matrix-by-vector multiplication. We now present an alternative design, based on mesh routing. This design performs a single routing operation per multiplication, compared to three sorting operations (where even a single sorting operation is slower than routing). The resulting design has a reduced cost, improved fault tolerance and very simple local control. Moreover, its inherent flexibility allows further improvements, as discussed in the next section. The basic design is as follows.

For simplicity assume that each of the  $D$  columns of the matrix has weight exactly  $h$  (here  $h = 100$ ), and that the nonzero elements of  $A$  are uniformly distributed (both assumptions can be easily relaxed). Let  $m = \sqrt{D \cdot h}$ . We divide the  $m \times m$  mesh into  $D$  blocks of size  $\sqrt{h} \times \sqrt{h}$ . Let  $S_i$  denote the  $i$ -th block in row-major order ( $i \in \{1, \dots, D\}$ ), and let  $t_i$  denote the node in the upper left corner of  $S_i$ . We say that  $t_i$  is the *target of the value  $i$* . Each node holds two  $\log_2 D$ -bit values,  $Q[i]$  and  $R[i]$ . Each target node  $t_i$  also contains a single-bit value  $P[i]$ . For repeated multiplication of  $A$  and  $\mathbf{v}$ , the mesh is initialized as follows: the  $i$ -th entry of  $\mathbf{v}$  is loaded into  $P[i]$ , and the row indices of the nonzero elements in column  $i \in \{1, \dots, D\}$  of  $A$  are stored (in arbitrary order) in the  $Q[\cdot]$  of the nodes in  $S_i$ . Each multiplication is performed thus:

1. For all  $i$ , broadcast the value of  $P[i]$  from  $t_i$  to the rest of the nodes in  $S_i$  (this can be accomplished in  $2\sqrt{h} - 2$  steps).
2. For all  $i$  and every node  $j$  in  $S_i$ : if  $P[i] = 1$  then  $R[j] \leftarrow Q[j]$ , else  $R[j] \leftarrow \text{nil}$  (where nil is some distinguished value outside  $\{1, \dots, D\}$ ).
3.  $P[i] \leftarrow 0$  for all  $i$
4. Invoke a mesh-based packet routing algorithm on the  $R[\cdot]$ , such that each non-nil value  $R[j]$  is routed to its target node  $t_{R[j]}$ . Each time a value  $i$  arrives at its target  $t_i$ , discard it and flip  $P[i]$ .

After these steps,  $P[\cdot]$  contain the result of the multiplication, and the mesh is ready for the next multiplication. As before, in the inner product computation stage of the Wiedemann algorithm, we need only compute  $\mathbf{u}A^k\mathbf{v}$  for some vector  $\mathbf{u}$ , so we load the  $i$ -th coordinate of  $\mathbf{u}$  into node  $t_i$  during initialization, and compute the single-bit result  $\mathbf{u}A^k\mathbf{v}$  inside the mesh during the next multiplication.

There remains the choice of a routing algorithm. Many candidates exist (see [7] for a survey). To minimize hardware cost, we restrict our attention to algorithms for the “one packet” model, in which at each step every node holds at most one packet (and consequentially each node can send at most one packet and receive at most one packet per step). Note that this rules out most known algorithms, including those for the well-studied “hot-potato” routing model which provides a register for every edge. Since we do binary multiplication, the routing problem has the following unusual property: pairwise packet annihilation is allowed. That is, pairs of packets with identical values may be “cancelled out” without affecting the result of the computation. This relaxation can greatly reduce the congestion caused by multiple packets converging to a common destination. Indeed this seems to render commonly-cited lower bounds inapplicable, and we are not aware of any discussion of this variant in the literature. While

known routing and sorting algorithms can be adapted to our task, we suggest a new routing algorithm that seems optimal, based on our empirical tests.

The algorithm, which we call *clockwise transposition routing*, has an exceptionally simple control structure which consists of repeating 4 steps. Each step involves compare-exchange operations on pairs of neighboring nodes, such that the exchange is performed iff it reduces the distance-to-target of the non-nil value (out of at most 2) that is farthest from its target along the relevant direction. This boils down to comparison of the target row indices (for vertically adjacent nodes) or target column indices (for horizontally adjacent nodes). For instance, for horizontally adjacent nodes  $i, i+1$  such that  $t_{R[i]}$  resides on column  $c_i$  and  $t_{R[i+1]}$  resides on column  $c_{i+1}$ , an exchange of  $i$  and  $i+1$  will be done iff  $c_i > c_{i+1}$ . To this we add annihilation: if  $R[i] = R[i+1]$  then both are replaced by nil.

The first step of clockwise transposition routing consists of compare-exchange between each node residing on an odd row with the node above it (if any). The second step consists of compare-exchange between each node residing on an odd column with the node to its right (if any). The third and fourth steps are similar to the first and second respectively, except that they involve the neighbors in the opposite direction. It is easily seen that each node simply performs compare-exchanges with its four neighbors in either clockwise or counterclockwise order.

We do not yet have a theoretical analysis of this algorithm. However, we have simulated it on numerous inputs of sizes up to  $13\,000 \times 13\,000$  with random inputs drawn from a distribution mimicking that of the above mesh, as well as the simple distribution that puts a random value in every node. In all runs (except for very small meshes), we have not observed even a single case where the running time exceeded  $2m$  steps. This is just two steps from the trivial lower bound  $2m - 2$ .

Our algorithm is a generalization of odd-even transposition sort, with a schedule that is identical to the “2D-bubblesort” algorithm of [8] but with different compare-exchange elements. The change from sorting to routing is indeed quite beneficial, as [8] shows that 2D-bubblesort is considerably slower than the observed performance of our clockwise transposition routing. The new algorithm appears to be much faster than the  $8m$  sorting algorithm (due to Schimmler) used in [1], and its local control is very simple compared to the complicated recursive algorithms that achieve the  $3m$ -step lower bound on mesh sorting (cf. [16]).

A physical realization of the mesh will contain many local faults (especially for devices that are wafer-scale or larger, as discussed below). In the routing-based mesh, we can handle local defects by algorithmic means as follows. Each node shall contain 4 additional state bits, indicating whether each of its 4 neighbors is “disabled”. These bits are loaded during device initialization, after mapping out the defects. The compare-exchange logic is augmented such that if node  $i$  has a “disabled” neighbor in direction  $\Delta$  then  $i$  never performs an exchange in that direction, but always performs the exchange in the two directions orthogonal to  $\Delta$ . This allows us to “close off” arbitrary rectangular regions of the mesh, such that values that reach a “closed-off” region from outside are routed along

its perimeter. We add a few spare nodes to the mesh, and manipulate the mesh inputs such that the spare effectively replace the nodes of the in closed-off regions. We conjecture that the local disturbance caused by a few small closed-off regions will not have a significant effect on the routing performance.

Going back to the cost evaluation, we see that replacing the sorting-based mesh with a routing-based mesh reduces time by a factor of  $3 \cdot 8/2 = 12$ . Also, note that the  $Q[\cdot]$  values are used just once per multiplication, and can thus be stored in slower DRAM cells in the vicinity of the node. DRAM cells are much smaller than edge-triggered flip-flops, since they require only one transistor and one capacitor per bit. Moreover, the regular structure of DRAM banks allows for very dense packing. Using large banks of embedded DRAM (which are shared by many nodes in their vicinity), the amortized chip area per DRAM bit is about  $0.7\mu\text{m}^2$ . Our Northwood-based estimates lead to  $2.38\mu\text{m}^2$  per transistor, so we surmise that for our purposes a DRAM bit costs  $1/3.4$  as much as a logic transistor, or about  $1/27$  as much as a flip-flop. For simplicity, we ignore the circuitry needed to retrieve the values from DRAM — this can be done cheaply by temporarily wiring chains of adjacent  $R[\cdot]$  into shift registers. In terms of circuit size, we effectively eliminate two of the three large registers per node, and some associated logic, so the routing-based mesh is about 3 times cheaper to manufacture. Overall, we gain a reduction of a factor  $3 \cdot 12 = 36$  in the throughput cost.

**5.5 An Improved Routing-Based Circuit.** We now tweak the routing-based circuit design to gain additional cost reductions. Compared to the sorting-based design (cf. [5.3](#)), these will yield a (constant-factor) improvement by several order of magnitudes. While asymptotically insignificant, this suggests a very practical device for the NFS matrix step of 1024-bit moduli. Moreover, it shows that already for 1024-bit moduli, the cost of parallelization can be negligible compared to the cost of the RAM needed to store the input, and thus the speed advantage is gained essentially for free.

The first improvement follows from increasing the density of targets. Let  $\rho$  denote the average number of  $P[\cdot]$  registers per node. In the above scheme,  $\rho = h^{-1} \approx 1/100$ . The total number of  $P[\cdot]$  registers is fixed at  $D$ , so if we increase  $\rho$  the number of mesh nodes decreases by  $h\rho$ . However, we no longer have enough mesh nodes to route all the  $hD$  nonzero entries of  $A$  simultaneously. We address this by partially serializing the routing process, as follows. Instead of storing one matrix entry  $Q[\cdot]$  per node, we store  $h\rho$  such values per node: for  $\rho \geq 1$ , each node  $j$  is “in charge” of a set of  $\rho$  matrix columns  $C_j = \{c_{j,1}, \dots, c_{j,\rho}\}$ , in the sense that node  $j$  contains the registers  $P[c_{j,1}], \dots, P[c_{j,\rho}]$ , and the nonzero elements of  $A$  in columns  $c_{j,1}, \dots, c_{j,\rho}$ . To carry out a multiplication we perform  $h\rho$  iterations, where each iteration consists of retrieving the next such nonzero element (or skipping it, depending on the result of the previous multiplication) and then performing clockwise transposition routing as before.

The second improvement follows from using block Wiedemann with a blocking factor  $K > 1$  (cf. [2.5](#)). Besides reducing the number of multiplications by a factor of roughly  $\frac{20}{3}$  (cf. [5.3](#)), this produces an opportunity for reducing the cost

of multiplication, as follows. Recall that in block Wiedemann, we need to perform  $K$  multiplication chains of the form  $A^k \mathbf{v}_i$ , for  $i = 1, \dots, K$  and  $k = 1, \dots, 2D/K$ , and later again, for  $k = 1, \dots, D/K$ . The idea is to perform several chains in parallel on a single mesh, reusing most resources (in particular, the storage taken by  $A$ ). For simplicity, we will consider handling all  $K$  chains on one mesh. In the routing-based circuits described so far, each node emitted at most one message per routing operation — a matrix row index, which implies the address of the target cell. The information content of this message (or its absence) is a single bit. Consider attaching  $K$  bits of information to this message:  $\log_2(D)$  bits for the row index, and  $K$  bits of “payload”, one bit per multiplication chain.

Combining the two generalizations gives the following algorithm, for  $0 < \rho \leq 1$  and integer  $K \geq 1$ . The case  $0 < \rho < 1$  requires distributing the entries of each matrix column among several mesh nodes, as in [5.4](#), but its cost is similar.

Let  $\{C_j\}_{j \in \{1, \dots, D/\rho\}}$  be a partition of  $\{1, \dots, D\}$ ,  $C_j = \{c : (j-1)\rho \leq c-1 < j\rho\}$ . Each node  $j \in \{1, \dots, D/\rho\}$  contains single-bit registers  $P_i[c]$  and  $P'_i[c]$  for all  $i = 1, \dots, K$  and  $c \in C_j$ , and a register  $R_j$  of size  $\log_2(D) + K$ . Node  $j$  also contains a list  $Q_j = \{(r, c) \mid A_{r,c} = 1, c \in C_j\}$  of the nonzero matrix entries in the columns  $C_j$  of  $A$ , and an index  $I_j$  into  $C_j$ . Initially, load the vectors  $\mathbf{v}_i$  into the  $P_i[\cdot]$  registers. Each multiplication is then performed thus:

1. For all  $i$  and  $c$ ,  $P'_i[c] \leftarrow 0$ . For all  $j$ ,  $I_j \leftarrow 1$ .
2. Repeat  $h\rho$  times:
  - (a) For all  $j$ :  $(r, c) \leftarrow Q_j[I_j]$ ,  $I_j \leftarrow I_j + 1$ ,  $R[j] \leftarrow \langle r, P_1[c], \dots, P_K[c] \rangle$ .
  - (b) Invoke the clockwise transposition routing algorithm on the  $R[\cdot]$ , such that each value  $R[j] = \langle r, \dots \rangle$  is routed to the node  $t_j$  for which  $r \in C_j$ . During routing, whenever a node  $j$  receives a message  $\langle r, p_1, \dots, p_K \rangle$  such that  $r \in C_j$ , it sets  $P'_i[r] \leftarrow P'_i[r] \oplus p_i$  for  $i = 1, \dots, K$  and discards the message. Moreover, whenever packets  $\langle r, p_1, \dots, p_K \rangle$  and  $\langle r, p'_1, \dots, p'_K \rangle$  in adjacent nodes are compared, they are combined: one is annihilated and the other is replaced by  $\langle r, p_1 \oplus p'_1, \dots, p_K \oplus p'_K \rangle$ .
3.  $P_i[c] \leftarrow P'_i[c]$  for all  $i$  and  $c$ .

After these steps,  $P_i[\cdot]$  contain the bits of  $A^k \mathbf{v}_i$  and the mesh is ready for the next multiplication. We need to compute and output the inner products  $\mathbf{u}_j(A^k \mathbf{v}_i)$  for some vectors  $\mathbf{u}_1, \dots, \mathbf{u}_K$ , and this computation should be completed before the next multiplication is done. In general, this seems to require  $\Theta(K^2)$  additional wires between neighboring mesh nodes and additional registers. However, usually the  $\mathbf{u}_j$  are chosen to have weight 1 or 2, so the cost of computing these inner products can be kept very low. Also, note that the number of routed messages is now doubled, because previously only half the nodes sent non-nil messages. However, empirically it appears that the clockwise transposition routing algorithm handles the full load without any slowdown.

It remains to determine the optimal values of  $K$  and  $\rho$ . This involves implementation details and technological quirks, and obtaining precise figures appears rather hard. We thus derive expressions for the various cost measures, based on parameters which can characterize a wide range of implementations. We then



substitute values that reasonably represent today’s technology, and optimize for these. The parameters are as follows:

- Let  $\mathcal{A}_t$ ,  $\mathcal{A}_f$  and  $\mathcal{A}_d$  be the average wafer area occupied by a logic transistor, an edge-triggered flip-flop and a DRAM bit, respectively (including the related wires).
- Let  $\mathcal{A}_w$  be the area of a wafer.
- Let  $\mathcal{A}_p$  be the wafer area occupied by an inter-wafer connection pad (cf. 5.3).
- Let  $\mathcal{C}_w$  be the construction cost of a single wafer (in large quantities).
- Let  $\mathcal{C}_d$  be the cost of a DRAM bit that is stored off the wafers (this is relevant only to the FPGA design of Appendix A).
- Let  $\mathcal{T}_d$  be the reciprocal of the memory DRAM access bandwidth of a single wafer (relevant only to FPGA).
- Let  $\mathcal{T}_l$  be the time it takes for signals to propagate through a length of circuitry (averaged over logic, wires, etc.).
- Let  $\mathcal{T}_p$  be the time it takes to transmit one bit through a wafer I/O pad.

We consider three implementation approaches: custom-produced “logic” wafers (as used in 5.3, with which we maintain consistency), custom-produced “DRAM” wafers (which reduce the size of DRAM cells at the expense of size and speed of logic transistors) and an FPGA-based design using off-the-shelf parts (cf. Appendix A). Rough estimates of the respective parameters are given in Table 2.

The cost of the matrix step is derived with some additional approximations:

- The number of mesh nodes is  $D/\rho$ .
- The values in  $Q_j[\cdot]$  (i.e., the nonzero entries of  $A$ ) can be stored in DRAM banks in the vicinity of the nodes, where (with an efficient representation) they occupy  $h\rho \log_2(D)\mathcal{A}_d$  per node.
- The  $P_i[c]$  registers can be moved to DRAM banks, where they occupy  $\rho K\mathcal{A}_d$  per node.
- The  $P'_j[c]$  registers can also be moved to DRAM. However, to update the DRAM when a message is received we need additional storage. Throughout the  $D/\rho$  steps of a routing operation, each node gets 1 message on average (or

**Table 2.** Implementation hardware parameters

	Custom1 (“logic”)	Custom2 (“DRAM”)	FPGA
$\mathcal{A}_t$	$2.38 \mu\text{m}^2$	$2.80 \mu\text{m}^2$	0.05
$\mathcal{A}_f$	$19.00 \mu\text{m}^2$	$22.40 \mu\text{m}^2$	1.00
$\mathcal{A}_d$	$0.70 \mu\text{m}^2$	$0.20 \mu\text{m}^2$	$\emptyset$
$\mathcal{A}_p$	$4\,000 \mu\text{m}^2 \times \text{sec}$	$4\,000 \mu\text{m}^2 \times \text{sec}$	$\emptyset$
$\mathcal{A}_w$	$6.36 \cdot 10^{10} \mu\text{m}^2$	$6.36 \cdot 10^{10} \mu\text{m}^2$	25 660
$\mathcal{C}_w$	\$5,000	\$5,000	\$150
$\mathcal{C}_d$	$\emptyset$	$\emptyset$	$\$4 \cdot 10^{-8}$
$\mathcal{T}_d$	$\emptyset$	$\emptyset$	$1.1 \cdot 10^{-11} \text{ sec}$
$\mathcal{T}_p$	$4 \cdot 10^{-9} \text{ sec}$	$4 \cdot 10^{-9} \text{ sec}$	$2.5 \cdot 10^{-9} \text{ sec}$
$\mathcal{T}_l$	$1.46 \cdot 10^{-11} \text{ sec}/\mu\text{m}$	$1.80 \cdot 10^{-11} \text{ sec}/\mu\text{m}$	$1.43 \cdot 10^{-9} \text{ sec}$

$\emptyset$  marks values that are inapplicable, and taken to be zero.



less, due to annihilation). Thus  $\log_2(\rho) + K$  latch bits per node would suffice (if they are still in use when another message arrives, it can be forwarded to another node and handled when it arrives again). This occupies  $\rho K \mathcal{A}_f$  per node when  $\rho < 2$ , and  $\rho K \mathcal{A}_d + 2(\log_2(\rho) + K) \mathcal{A}_f$  per node when  $\rho \geq 2$ .

- The bitwise logic related to the  $P_i[c]$  registers, the  $P'_i[c]$  and the last  $K$  bits of the  $R[j]$  registers together occupy  $20 \cdot \min(\rho, 2) K \mathcal{A}_t$  per node.
- The  $R[j]$  registers occupy  $(\log_2(D) + K) \mathcal{A}_f$  per node
- The rest of the mesh circuitry (clock distribution, DRAM access, clockwise transposition routing, I/O handling, inner products, etc.) occupies  $(200 + 30 \log_2(D)) \mathcal{A}_t$  per node.
- Let  $\mathcal{A}_n$  be total area of a mesh node, obtained by summing the above (we get different formulas for  $\rho < 2$  vs.  $\rho \geq 2$ ).
- Let  $\mathcal{A}_m = \mathcal{A}_n D / \rho$  be the total area of the mesh nodes (excluding inter-wafer connections).
- Let  $\mathcal{N}_w$  be the number of wafers required to implement the matrix step, and let  $\mathcal{N}_p$  be the number of inter-wafer connection pads per wafer. For single-wafer designs,  $\mathcal{N}_w = 1 / \lfloor \mathcal{A}_w / \mathcal{A}_m \rfloor$  and  $\mathcal{N}_p = 0$ . For multiple-wafer designs, these values are derived from equations for wafer area and bandwidth:  $\mathcal{N}_w \mathcal{A}_w = \mathcal{A}_m + \mathcal{N}_w \mathcal{N}_p \mathcal{A}_p$ ,  $\mathcal{N}_p = 4 \cdot 2 \cdot \sqrt{D / (\rho \mathcal{N}_w)} \cdot (\log_2 D + K) \cdot \mathcal{T}_p / (\sqrt{\mathcal{A}_n} \mathcal{T}_l)$ .
- Let  $\mathcal{N}_d$  be total number of DRAM bits (obtained by evaluating  $\mathcal{A}_m$  for  $\mathcal{A}_f = \mathcal{A}_t = 0, \mathcal{A}_d = 1$ ).
- Let  $\mathcal{N}_a$  be the number of DRAM bit accesses (reads+writes) performed throughout the matrix step. We get:  $\mathcal{N}_a = 3D(2hDK + Dh \log_2(D))$ , where the first term due to the  $P'_i[c]$  updates and the second term accounts for reading the matrix entries.
- Let  $\mathcal{C}_s = \mathcal{N}_w \mathcal{C}_w + \mathcal{N}_d \mathcal{C}_d$  be the total construction cost for the matrix step.
- The full block Wiedemann algorithm consists of  $3D/K$  matrix-by-vector multiplications, each of which consists of  $h\rho$  routing operations, each of which consists of  $2\sqrt{D/\rho}$  clocks. Each clock cycle takes  $\mathcal{T}_l \sqrt{\mathcal{A}_n}$ .  
Let  $\mathcal{T}_s$  be the time taken by the full block Wiedemann algorithm. We get:  
 $\mathcal{T}_s = 6D^{3/2} h \mathcal{T}_l \sqrt{\rho \mathcal{A}_n} / K + \mathcal{N}_a \mathcal{T}_d / \mathcal{N}_w$ .

Table 3 lists the cost of the improved routing-based circuit for several choices of  $\rho$  and  $K$ , according to the above. It also lists the cost of the sorting-based circuits (cf. 5.3) and the PC implementation of Appendix B. The lines marked by “(opt)” give the parameter choice that minimize the throughput cost for each type of hardware.

The second line describes a routing-based design whose throughput cost is roughly 45 000 times lower than that of the original sorting-based circuit (or 6 700 times lower than sorting with  $K \gg 1$ ). Notably, this is a single-wafer device, which completely solves the technological problem of connecting multiple wafers with millions of parallel wires, as necessary in the original design of [1]. The third line shows that significant parallelism can be gained essentially for free: here, 88% of the wafer area is occupied simply by the DRAM banks needed to store the input matrix, so further reduction in construction cost seems impossible.

**Table 3.** Cost of the matrix step for the “small” matrix

Algorithm	Implementation	$\rho$	$K$	Wafers/ chips/ PCs	Construction cost $C_s$	Run time $T_s$ (sec)	Throughput cost $C_s T_s$ (\$ $\times$ sec)
Routing	Custom1	0.51	107	19	\$94,600	1440 (24 min)	$1.36 \cdot 10^8$ (opt)
Routing	Custom2	42.10	208	1	\$5,000	21 900 (6.1 hours)	$1.10 \cdot 10^8$ (opt)
Routing	Custom2	216.16	42	0.37	\$2,500	341 000 (4 days)	$8.53 \cdot 10^8$
Routing	Custom1	0.11	532	288	\$1,440,000	180 (3 min)	$2.60 \cdot 10^8$
Routing	FPGA	5473.24	25	64	\$13,800	15 900 000 (184 days)	$2.20 \cdot 10^{11}$ (opt)
Routing	FPGA	243.35	60	2500	\$380,000	1 420 000 (17 days)	$5.40 \cdot 10^{11}$
Sorting	Custom1		1	273	\$4,100,000	1 210 000 (14 days)	$4.96 \cdot 10^{12}$
Sorting	Custom1		$\gg 1$	273	\$4,100,000	182 000 (50 hours)	$7.44 \cdot 10^{11}$
Serial	PCs		32	1	\$4,460	125 000 000 (4 years)	$5.59 \cdot 10^{11}$
Tree	PCs		32	66	\$24,000	2 290 000 (27 days)	$5.52 \cdot 10^{10}$

**Table 4.** Cost of the matrix step for the “large” matrix

Algorithm	Implementation	$\rho$	$K$	Wafers/ chips/ PCs	Construction cost $C_s$	Run time $T_s$ (sec)	Throughput cost $C_s T_s$ (\$ $\times$ sec)
Routing	Custom1	0.51	136	6030	\$30.1M	$5.04 \cdot 10^6$ (58 days)	$1.52 \cdot 10^{14}$ (opt)
Routing	Custom2	4112	306	391	\$2.0M	$6.87 \cdot 10^7$ (2.2 years)	$1.34 \cdot 10^{14}$ (opt)
Routing	Custom2	261.60	52	120	\$0.6M	$1.49 \cdot 10^9$ (47 years)	$8.95 \cdot 10^{14}$
Routing	Custom1	0.11	663	9000	\$500.0M	$6.40 \cdot 10^5$ (7.4 days)	$2.88 \cdot 10^{14}$
Routing	FPGA	17 757.70	99	13 567	\$3.5M	$3.44 \cdot 10^{10}$ (1088 years)	$1.19 \cdot 10^{17}$ (opt)
Routing	FPGA	144 .41	471	$6.6 \cdot 10^6$	\$1000.0M	$1.14 \cdot 10^9$ (36 years)	$1.13 \cdot 10^{18}$
Serial	PCs		32	1	\$1.3M	270 000 years	$1.16 \cdot 10^{19}$
Tree	PCs		3484	813	\$153.0M	$3.17 \cdot 10^8$ (10 years)	$4.84 \cdot 10^{16}$

**5.6 An Improved Circuit for the “Large” Matrix.** The large matrix resulting from ordinary relation collection contains 250 times more columns:  $D \approx 10^{10}$ . We assume that the average column density remains  $h = 100$ . It is no longer possible to fit the device on a single wafer, so the feasibility of the mesh design now depends critically on the ability to make high bandwidth inter-wafer connections (cf. [5.3](#)).

Using the formulas given in the previous section, we obtain the costs in [Table 4](#) for the custom and FPGA implementations, for various parameter choices. The third line shows that here too, significant parallelism can be attained at very little cost (88% of the wafer area is occupied by DRAM storing the input). As can be seen, the improved mesh is quite feasible also for the large matrix, and its cost is a small fraction of the cost of the alternatives, and of relation collection.

**5.7 Summary of Hardware Findings.** The improved design of [5.5](#) and [5.6](#), when implemented using custom hardware, appears feasible for both matrix sizes. Moreover, it is very attractive when compared to the traditional serial implementations (though appropriate parallelization techniques partially close this gap; see [Appendix B](#)). However, these conclusions are based on numerous assumptions, some quite optimistic. Much more research, and possibly actual relation collection experiments, would have to be carried out to get a clearer

grasp of the actual cost (time and money) of both the relation collection and matrix steps for 1024-bit moduli.

In light of the above, one may try to improve the overall performance of NFS by re-balancing the relation collection step and the matrix step, i.e., by increasing the smoothness bounds (the opposite of the approach sketched in Remark 3.7). For ordinary NFS, asymptotically this is impossible since the parameters used for ordinary relation collection (i.e., the “large” matrix) already minimize the cost of relation collection (cf. 2.6). For improved NFS that is applied to a single factorization (cf. 2.7), if we disregard the cost of the matrix step and optimize just for relation collection then we can expect a cost reduction of about  $L_{2^{1024}}[1/3, 1.9018836 \dots] / L_{2^{1024}}[1/3, 1.8689328 \dots] \approx 2.8$ .

If many integers in a large range must be factored — a reasonable assumption given our interpretation of the throughput cost (cf. 3.2) — a much faster method exists (cf. 4). It remains to be studied whether these asymptotic properties indeed hold for 1024-bit moduli and what are the practical implications of the methods from 4.

## 6 Conclusion

We conclude that methods to evaluate the security of RSA moduli that are based on the traditional operation count are not affected by the circuits proposed in 1. Although the traditional estimates underestimate the difficulty of factoring, 1 provides yet another reason — other than the mostly historical reasons used so far — not to rely too much on supposedly more accurate cost-based estimates for the NFS.

We have shown that the suggestion made in 1 that the number of digits of factorable numbers has grown by a factor of 3, is based on an argument that may not be to everyone’s taste. An alternative interpretation leads to a factor 1.17, under the cost function defined in 1. The most traditional cost function, however, even leads to a factor 0.92.

Finally, we have presented an improved design for a mesh-based implementation of the linear algebra stage of the NFS. For an optimistically estimated 1024-bit factorization, our analysis suggests that a linear dependency between the columns of the sparse matrix can be found within a few hours by a device that costs about \$5,000. At the very least, this is an additional argument not to rely on the alleged difficulty of the matrix step when evaluating the difficulty of factoring. As mentioned in 1 there are many other possibilities to be explored. Further study — and unbiased interpretation of the results — should eventually enable the cryptographic research and users communities to assess the true impact of 1 and the method proposed in 5.5.

## Acknowledgments

We thank Daniel J. Bernstein for his constructive criticism 2; we believe that these concerns are addressed by the present paper. We thank Martijn Stam for

his assistance with the formulas in [2.7], Scott Contini for his careful reading and his insightful comments, and Yuliang Zheng for his kind cooperation. The first author thanks John Markoff for bringing [1] to his attention. The third author worked at Citibank when the first version of this paper was written.

## References

1. D.J. Bernstein, *Circuits for integer factorization: a proposal*, manuscript, November 2001; available at [cr.yp.to/papers.html#nfscircuit](http://cr.yp.to/papers.html#nfscircuit)
2. D.J. Bernstein, *Circuits for integer factorization*, web page, July 2002; <http://cr.yp.to/nfscircuit.html>
3. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, et al., *Factorization of a 512-bit RSA modulus*, Proceedings Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 1-17
4. D. Coppersmith, *Modifications to the number field sieve*, Journal of Cryptology **6** (1993) 169-180
5. D. Coppersmith, *Solving homogeneous linear equations over  $GF(2)$  via block Wiedemann algorithm*, Math. Comp. bf 62 (1994) 333-350
6. B. Dixon, A.K. Lenstra, *Factoring integers using SIMD sieves*, Proceedings Eurocrypt 1993, LNCS 765, Springer-Verlag 1994, 28-39
7. M. D. Grammatikakis, D. F. Hsu, M. Kraetzl, J. F. Sibeyn, *Packet routing in fixed-connection networks: a survey*, Journal of Parallel and Distributed Computing, 54(2):77-132, Nov. 1998
8. D. Ierardi, *2d-Bubblesorting in average time  $O(N \lg N)$* , Proceedings 6th ACM symposium on Parallel algorithms and architectures, 1994
9. A.K. Lenstra, *Unbelievable security; matching AES security using public key systems*, Proceedings Asiacrypt 2001, LNCS 2248, Springer-Verlag 2001, 67-86
10. A.K. Lenstra, H.W. Lenstra, Jr., *Algorithms in number theory*, chapter 12 in *Handbook of theoretical computer science, Volume A, algorithms and complexity* (J. van Leeuwen, ed.), Elsevier, Amsterdam (1990)
11. A.K. Lenstra, H.W. Lenstra, Jr., (eds.), *The development of the number field sieve*, Lecture Notes in Math. **1554**, Springer-Verlag 1993
12. A.K. Lenstra, E.R. Verheul, *Selecting cryptographic key sizes*, J. of Cryptology, **14** (2001) 255-293; available at [www.cryptosavvy.com](http://www.cryptosavvy.com)
13. P.L. Montgomery, *A block Lanczos algorithm for finding dependencies over  $GF(2)$* , Proceedings Eurocrypt'95, LNCS 925, Springer-Verlag 1995, 106-120
14. NIST, *Key management guideline – workshop document*, Draft, October 2001; available at [csrc.nist.gov/encryption/kms](http://csrc.nist.gov/encryption/kms)
15. R.D. Silverman, *A cost-based security analysis of symmetric and asymmetric key lengths*, Bulletin 13, RSA laboratories, 2000; available at [www.rsasecurity.com/rsalabs/bulletins/index.html](http://www.rsasecurity.com/rsalabs/bulletins/index.html)
16. C.P. Schnorr, A. Shamir, *An Optimal Sorting Algorithm for Mesh Connected Computers*, Proceedings 16th ACM Symposium on Theory of Computing, 255-263, 1986
17. G. Villard, *Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems (extended abstract)*, Proceedings 1997 International Symposium on Symbolic and Algebraic Computation, ACM Press, 32-39, 1997

18. D. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Transactions on Information Theory, **IT-32** (1986), 54–62
19. M.J. Wiener, *The full cost of cryptanalytic attacks*, accepted for publication in J. of Cryptology

## A Using Off-the-Shelf Hardware for the Circuit Approach

In subsections 5.3–5.5 we were concerned primarily with custom-produced hardware, in accordance with the focus on throughput cost. In practice, however, we are often concerned about solving a small number of factorization problems. In this case, it may be preferable to use off-the-shelf components (especially if they can be dismantled and reused, or if discreteness is desired).

Tables 2–4 in Section 5.5 contain the parameters and cost estimates for off-the-shelf hardware, using the following scheme. FPGA chips are connected in a two-dimensional grid, where each chip holds a block of mesh nodes. The FPGA we consider is the Altera Stratix EP1S25F1020C7, which is expected to cost about \$150 in large quantities in mid-2003. It contains 2Mbit of DRAM and 25 660 “logic elements” that consist each of a single-bit register and some configurable logic. Since on-chip DRAM is scant, we connect each FPGA to several DRAM chips. The FPGA has 706 I/O pins that can provide about 70Gbit/sec of bandwidth to the DRAM chips (we can fully utilize this bandwidth by “swapping” large continuous chunks into the on-FPGA DRAM; the algorithm allows efficient scheduling). These I/O pins can also be used for communicating with neighbouring FPGAs at an aggregate bandwidth of 280Gbit/sec.

The parameters given in Table 2 are normalized, such that one LE is considered to occupy 1 area unit, and thus  $\mathcal{A}_f = 1$ . We make the crude assumption that each LE provides the equivalent of 20 logic transistors in our custom design, so  $\mathcal{A}_t = 0.05$ . Every FPGA chip is considered a “wafer” for the purpose of calculation, so  $\mathcal{A}_w = 51\,840$ . Since DRAM is located outside the FPGA chips,  $\mathcal{A}_d = 0$  but  $\mathcal{C}_d = 4 \cdot 10^8$ , assuming \$320 per gigabyte of DRAM.  $\mathcal{T}_d$  and  $\mathcal{T}_p$  are set according to available bandwidth. For  $\mathcal{T}_l$  we assume that on average an LE switches at 700MHz.  $\mathcal{A}_p = 0$ , but we need to verify that the derived  $\mathcal{N}_p$  is at most 706 (fortunately this holds for all our parameter choices).

As can be seen from the tables, the FPGA-based devices are significantly less efficient than both the custom designs and properly parallelized PC-based implementation. Thus they appear unattractive.

## B The Traditional Approach to the Matrix Step

We give a rough estimate of the price and performance of a traditional implementation of the matrix step using the block Lanczos method [13] running on standard PC hardware. Let the “small” and “large” matrices be as in 5.1

**B.1 Processing the “Small” Matrix Using PCs.** A bare-bones PC with a 2GHz Pentium 4 CPU can be bought for \$300, plus \$320 per gigabyte of RAM. We will use block Lanczos with a blocking factor of 32, to match the processor word size. The  $hD = 4 \cdot 10^9$  nonzero entries of the “small” matrix require 13GB

of storage, and the auxiliary  $D$ -dimensional vectors require under 1GB. The construction cost is thus about \$4,500.

The bandwidth of the fastest PC memory is 4.2GB/sec. In each matrix-by-vector multiplication, all the nonzero matrix entries are read, and each of these causes an update (read and write) of a 32-bit word. Thus, a full multiplication consists of accessing  $hD \log_2(D) + 2hD \cdot 32 = 4.8 \cdot 10^{10}$  bits, which takes about 11 seconds. The effect of the memory latency on non-sequential access, typically  $40n$ , raises this to about 50 seconds (some reduction may be possible by optimizing the memory access pattern to the specific DRAM modules used, but this appears nontrivial). Since  $2D/32$  matrix-by-vector multiplications have to be carried out [13], we arrive at a total of  $1.25 \cdot 10^8$  seconds (disregarding the cheaper inner products), i.e., about 4 years.

The throughput cost is  $5.6 \cdot 10^{11}$ , which is somewhat better than the sorting-based mesh design (despite the asymptotic advantage of the latter), but over 5000 times worse than the the single-wafer improved mesh design (cf. 5.5). Parallelization can be achieved by increasing the blocking factor of the Lanczos algorithm — this would allow for different tradeoffs between construction cost and running time, but would not decrease the throughput cost.

**B.2 Processing the “Large” Matrix Using PCs.** The large matrix contains 250 times more columns at the same (assumed) average density. Thus, it requires 250 times more memory and  $250^2 = 62\,500$  times more time than the small matrix. Moreover, all row indices now occupy  $\lceil \log_2 10^9 \rceil = 34$  bits instead of just 24. The cost of memory needed to store the matrix is \$1.36M (we ignore the lack of support for this amount of memory in existing memory controllers), and the running time is 270 000 years. This appears quite impractical (we cannot increase the blocking factor by over  $\sqrt{D}$ , and even if we could, the construction cost would be billions of dollars).

**Remark B.3.** Once attention is drawn to the cost of memory, it becomes evident that better schemes are available for parallelizing a PC-based implementation. One simple scheme involves distributing the matrix columns among numerous PCs such that each node  $j$  is in charge of some set of columns  $C_j \subset \{1, \dots, D\}$ , and contains only these matrix entries (rather than the whole matrix). The nodes are networked together with a binary tree topology. Let  $\mathbf{a}_i$  denote the  $i$ -th column of  $A$ . Each matrix-by-vector multiplication  $A\mathbf{w}$  consists of the root node broadcasting the bits  $w_1, \dots, w_D$  down the tree, each node  $j$  computing a partial sum vector  $\mathbf{r}_j = \sum_{i \in C_j, w_i=1} \mathbf{a}_i \pmod{2}$ , and finally performing a converge-cast operation to produce the sum  $\sum_j \mathbf{r}_j = A\mathbf{w} \pmod{2}$  at the root. If the broadcast and converge-cast are done in a pipelined manner on 0.5 gigabit links, this is easily seen to reduce the throughput cost to roughly  $5.6 \cdot 10^{10}$  for the small matrix and  $4.8 \cdot 10^{16}$  for the large matrix (see Tables 3.4).

For constant-bandwidth links, this scheme is asymptotically inefficient since its throughput cost is  $L(3\beta)$ . However, for the parameters considered it is outperformed only by the custom-built improved mesh.

# A Variant of the Cramer-Shoup Cryptosystem for Groups of Unknown Order

Stefan Lucks

Theoretische Informatik, Universität Mannheim, 68131 Mannheim, Germany,  
lucks@th.informatik.uni-mannheim.de

**Abstract.** The Cramer-Shoup cryptosystem for groups of prime order is a practical public-key cryptosystem, provably secure in the standard model under standard assumptions. This paper extends the cryptosystem for groups of unknown order, namely the group of quadratic residues modulo a composed  $N$ . Two security results are: In the standard model, the scheme is provably secure if both the Decisional Diffie-Hellman assumption for  $\text{QR}_N$  and the factorisation assumption for  $N$  hold. In the random oracle model, the scheme is provably secure under the factorisation assumption by a quite efficient reduction.

## 1 Introduction

Security against chosen ciphertext attacks is essential for many cryptosystems. Naor and Yung [11] introduced this notion into the world of public-key cryptosystems and first described a scheme secure against non-adaptive chosen ciphertext (“lunchtime”) attacks. Today, most cryptographers agree that a “good” public-key cryptosystem should be secure against *adaptive chosen ciphertext* (ACC) attacks.<sup>1</sup> This notion has been introduced by Rackoff and Simon [12]. Dolev, Dwork and Naor [9] described a scheme provably secure against ACC attacks under standard assumptions. However, their scheme is too inefficient for practical applications. The research for provably secure and practically efficient cryptosystems has led to schemes provably secure in the *random oracle model* [2], and to schemes provably secure under non-standard assumptions such as the “oracle Diffie-Hellman” assumption [1].

The Cramer-Shoup cryptosystem [5] is the only cryptosystem known to be *both* practical *and* provably secure under standard assumptions – mainly, the decisional Diffie-Hellman assumption in groups of prime order. Recently, the same authors proposed a generalisation of their cryptosystem [7]. Its security can be based either on Paillier’s decision composite residuosity assumption or on the (quite classical) quadratic residuosity (QR) assumption – or on the decisional Diffie-Hellman assumption in groups of prime order, as before. As pointed out in [7], the QR-based variant of the generalisation is not too efficient in practice.<sup>2</sup> In

<sup>1</sup> Some authors denote lunchtime attacks by “IND-CCA1” and ACC attacks by “IND-CCA2”.

<sup>2</sup> A sample instantiation of the security parameters with  $N \approx 2^{1024}$  in [7] implies the following: A public key needs 70 KB of storage space, and an encryption operation



this paper, we deal with another variation, based on the Diffie-Hellman problem in specific groups of non-prime order.

Set  $N = PQ$ ,  $P = 2p + 1$ ,  $Q = 2q + 1$ ,  $p \neq q$ , and let  $P$ ,  $Q$ ,  $p$ , and  $q$  be odd primes. In the remainder of this paper, we assume  $N$  to be of that form. Consider the group  $\text{QR}_N$  of the Quadratic Residues mod  $N$  and the Cramer-Shoup Cryptosystem in this group. ([5] originally proposed their cryptosystem for groups of prime order only.) As it will turn out, the legal user will not need to know the factorisation of  $N$  for either encryption, decryption or key generation (with the possible exception of generating an appropriate  $N$  itself). Since knowing the factorisation of  $N$  is equivalent to knowing the order of  $\text{QR}_N$ , the group  $\text{QR}_N$  may be of *unknown order* even for the legal user.

A security result in the standard model provides assurance against all attacks, while a random oracle security result only provides assurance against so-called “generic” attacks. On the other hand, it is desirable to base the security of cryptosystems on weak assumptions, instead of strong ones. In this spirit, Shoup [13] proposed a “hedged” variant of the Cramer-Shoup cryptosystem, being both provably secure in the standard model under a strong assumption *and* provably secure in the random oracle model under a weak assumption. In Section 7 we follow the same approach. Our extension is different from Shoup’s technique, and the proof for the security in the random oracle model given here is more efficient than its counterpart in [13].

## 2 Properties of the Set $\text{QR}_N$

In this section, we recall some number-theoretic terminology and facts. Let  $G$  be a finite multiplicative group of the order  $|G| \geq 2$ . The order  $\text{ord}(x)$  of  $x \in G$  is the smallest integer  $e > 0$  such that  $x^e = x^0$ .  $G$  is *cyclic*, if a *generator*  $g$  for  $G$  exists, i.e., an element  $g \in G$  with  $\text{ord}(x) = |G|$ . Further,  $\{1\}$  and  $G$  itself are the two *trivial* subgroups of  $G$ , all other subgroups are nontrivial.

Recall that  $N = PQ$ , where  $P = 2p + 1$ ,  $Q = 2q + 1$ ,  $p$  and  $q$  are primes (i.e., both  $p$  and  $q$  are Sophie-Germain primes). Consider the set  $\text{QR}_N = \{x \in \mathbb{Z}_N^* \mid \exists a \in \mathbb{Z}_N^* : a^2 \equiv x \pmod{N}\}$  of Quadratic Residues modulo  $N$ . In the sequel, we use the following lemmas, which we prove in Section A of the appendix.

**Lemma 1.**  *$\text{QR}_N$  has a nontrivial subgroup of order  $p$  and a nontrivial subgroup of order  $q$ . Both subgroups are cyclic.*

**Lemma 2.**  *$\text{QR}_N$  is cyclic. It consists of one element of the order 1,  $(p - 1)$  elements of the order  $p$ ,  $(q - 1)$  elements of the order  $q$ , and  $(p - 1)(q - 1)$  elements of the order  $pq$ .*

**Lemma 3.** *For every  $x \in \text{QR}_N$ :  $\text{ord}(x) \in \{p, q\} \Rightarrow \gcd(x - 1, N) \in \{P, Q\}$ .*

---

needs about 600 exponentiations modulo  $N$ . Note that the other variants are much more efficient.



**Lemma 4.** *Let  $g$  be a generator for  $QR_N$ . For every  $x \in \mathbb{Z}_{pq}$ :  $\text{ord}(g^x) \in \{p, q\} \Leftrightarrow \gcd(x, pq) \in \{p, q\}$ .*

Computations in  $QR_N$  are computations modulo  $N$ . If it is implied by context, we omit writing explicitly “mod  $N$ ” for calculations mod  $N$ . If  $S$  is a finite set, we write  $v \in_{\mathbb{R}} S$  if the value  $v$  is chosen from the set  $S$  according to the uniform probability distribution. We write  $x \in_{\mathbb{R}} \mathbb{Z}_{pq}$  for randomly choosing  $x$  in  $\mathbb{Z}_{pq}$  according to a distribution statistically indistinguishable from uniform. Consider, e.g.,  $x \in_{\mathbb{R}} \mathbb{Z}_{\lfloor N/4 \rfloor}$ . Since  $\lfloor N/4 \rfloor \leq pq + p/2 + q/2 + 1/4$ ,  $x \in \mathbb{Z}_{pq}$  is overwhelmingly probable:  $\Pr[x \in \mathbb{Z}_{pq}] \geq 1 - \frac{p+q}{2pq} \geq \min\{1 - \frac{1}{p}, 1 - \frac{1}{q}\}$ .

### 3 Key Encapsulation Mechanisms

A key encapsulation mechanism (KEM) can be seen as the *secret-key part* of a hybrid cryptosystem. Combining a KEM with an appropriate secret-key cryptosystem provides the functionality of a public-key cryptosystem. If the secret-key cryptosystem satisfies some fairly standard security assumptions and the KEM is secure against ACC attacks, the public-key cryptosystem is secure against ACC attacks as well. (This is called a “folk theorem” in [13]. See also [6].) A KEM is a triple (Gen, KE, KD) of algorithms:

1. A *key pair generation* algorithm Gen, which, given a security parameter, randomly chooses a public-key/secret-key pair (PK, SK).
2. A randomised *key encapsulation* algorithm KE to choose  $(C, K) = \text{KE}(\text{PK})$ , i.e. a ciphertext  $C$  and an encapsulated key  $K$ .
3. A deterministic *key decapsulation* algorithm KD to compute  $K' = \text{KD}(\text{SK}, C)$ , and to reject invalid ciphertexts.

A KEM is *sound*, if  $K = K'$  for any  $(\text{PK}, \text{SK}) = \text{Gen}(\cdot)$ ,  $(C, K) = \text{KE}(\text{PK})$ , and  $K' = \text{KD}(\text{SK}, C)$ . The KEM presented in Section 4 and its extension in Section 7 are both sound. Proving this is easy, but omitted here for the sake of space.

An *ACC attack* against a KEM (Gen, KE, KD) can be described by the following game:

1. A *key generation* oracle computes  $(\text{PK}, \text{SK}) = \text{Gen}(\cdot)$  and publishes PK.
2. A *key encapsulation* oracle chooses  $(C, K) = \text{KE}(\text{PK})$  and  $\sigma \in_{\mathbb{R}} \{0, 1\}$ . If  $\sigma = 0$ , the oracle sends  $(C, K)$  to the adversary, else  $(C, K')$  with  $K' \in_{\mathbb{R}} \{0, 1\}^{|K|}$ .
3. The adversary makes some queries  $C_1, \dots, C_q$  to a *key decapsulation* oracle, with  $C_i \neq C$ . For each query  $C_i$ , the oracle responds the value  $\text{KD}(\text{SK}, C_i)$ , which may be either a bit string, or a special code to indicate rejection. For  $i \in \{1, \dots, q-1\}$ , the adversary learns the response  $\text{KD}(\text{SK}, C_i)$  before she has to choose the next query  $C_{i+1}$ .
4. The adversary outputs a value  $\sigma' \in \{0, 1\}$ .

The *adversary's advantage* in guessing  $\sigma$  is the difference

$$|\Pr[\sigma' = 1 | \sigma = 1] - \Pr[\sigma' = 1 | \sigma = 0]|$$

of conditional probabilities. A KEM is *secure against ACC attacks*, or *ACC-secure* if, for all efficient adversaries, the advantage is negligible.

In Section [B](#) of the appendix, we compare ACC-secure KEMs with ACC-secure public-key cryptosystems and introduce lunchtime-security.

## 4 The Cryptosystem and Some Assumptions

Here, we deal with the Cramer-Shoup cryptosystem and what assumptions we make to prove its security. Cramer and Shoup [\[5\]](#) considered groups  $G$  of (known) prime order  $q^*$ , while we consider the group  $\text{QR}_N$  of composed order  $pq$ . There is no need to actually know  $pq$ , not even for the owner of the secret key. (Note that knowing  $pq$  makes factorising  $N$  easy.) For the sake of simplicity, we restrict ourselves to describing the system as a key encapsulation mechanism, instead of a full scale public-key cryptosystem.

### Cramer-Shoup Cryptosystem in the Group $\text{QR}_N$ :

- Key Generation  $\text{Gen}(l)$ :
  - Generate  $N$ ,  $P$ ,  $Q$ ,  $p$ ,  $q$  as above with  $2^{l-1} < N < 2^l$ .  
Choose a generator  $g$  for  $\text{QR}_N$ .
  - Choose a hash function  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_m$  (with  $m \leq pq$ ).
  - Randomly choose  $w \in_{\text{r}} \mathbb{Z}_{pq}$ , and compute  $g_2 = g^w$ . Choose  $x_1, x_2, y_1, y_2, z \in_{\text{r}} \mathbb{Z}_{pq}$ . Compute  $c = g^{x_1} g_2^{x_2}$ ,  $d = g^{y_1} g_2^{y_2}$ , and  $e = g^z$ .
  - The public key is  $\text{PK}=(N, g, H, g_2, c, d, e)$ .  
The secret key is  $\text{SK}=(x_1, x_2, y_1, y_2, z)$  in  $\mathbb{Z}_{pq}^5$ .
- Key Encapsulation  $\text{KE}(\text{PK})$ :
  - Choose  $r \in_{\text{r}} \mathbb{Z}_{pq}$ , compute  $u_1 = g^r$ ,  $u_2 = g_2^r$ ,  $k = e^r$ ,  $\alpha = H(u_1, u_2)$  and  $t = c^r d^{r\alpha}$ .
  - The ciphertext is  $(u_1, u_2, t)$ , the encapsulated key is  $k$ .
- Key Decapsulation [\[3\]](#)  $\text{KD}(\text{SK}, (U_1, U_2, T))$  for  $(U_1, U_2, T) \in \text{QR}_N^2 \times \mathbb{Z}_N^*$ :
  - Compute  $K' = U_1^z$ ,  $A' = H(U_1, U_2)$ ,  $T' = U_1^{x_1+y_1 A'} U_2^{x_2+y_2 A'}$ .
  - If  $T = T'$  then output  $K'$ , else reject.

Both in a group  $G$  of prime order and in composed order groups (such as  $\text{QR}_N$  and  $\mathbb{Z}_N^*$ ), expressions such as  $g^a * g^b$  and  $(g^a)^b$  are equivalent to  $g^{a+b}$  and  $g^{ab}$ . For prime order groups “ $a + b$ ” and “ $ab$ ” are addition and multiplication in a *field*, but for general groups  $G$  these operations are defined in the *ring*  $\mathbb{Z}_{|G|}$ . Thus, the proof of security from [\[5\]](#) is not directly applicable to the cryptosystem proposed in the current paper, though our proof is along the same lines.

<sup>3</sup> We don’t care if  $T \notin \text{QR}_N$ , because  $T' \in \text{QR}_N$ , and the test “ $T = T'$ ” is supposed to fail if  $T \notin \text{QR}_N$ . Remark [\[3\]](#) describes how to enforce  $U_1, U_2 \in \text{QR}_N$ .

**Assumption:** (Target collision resistance of  $H$ )

Let  $F_H$  be a family of hash functions  $\{0, 1\}^* \rightarrow \mathbb{Z}_m$ , for  $m \leq pq$ . Consider the following experiment:

1. Fix an input  $T$  for  $H$  (the “target”).
2. Randomly choose  $H$  from the family  $F_H$ .

It is infeasible to find a “collision” for the target  $T$ , i.e., an input  $T' \neq T$  such that  $H(T) = H(T')$ .

As a minor abuse of notation, we write “ $H$  is target collision resistant” (“TC-resistant”) to indicate that  $H$  has been chosen from such a family  $F_H$ .

**Assumption:** (decisional Diffie-Hellman (DDH) assumption for  $QR_N$ )

Let a generator  $g$  for  $QR_N$  be given. Consider the distributions  $R$  of triples  $(g_2, u_1, u_2) \in {}_{\mathbb{R}} QR_N^3$  and  $D$  of triples  $(g_2, u_1, u_2)$  with  $g_2 \in {}_{\mathbb{R}} QR_N$ ,  $r \in {}_{\mathbb{R}} \mathbb{Z}_{pq}$ ,  $u_1 = g^r$ , and  $u_2 = g_2^r$ . It is infeasible to distinguish between  $R$  and  $D$ . 4

**Assumption:** (computational Diffie-Hellman (CDH) assumption for  $QR_N$ )

Let a generator  $g$  for  $QR_N$  be given. Given two values  $g_2 \in {}_{\mathbb{R}} QR_N$  and  $u_1 \in {}_{\mathbb{R}} QR_N$  with  $\log_g(u_1) = r$ , it is infeasible to find the value  $u_2 = g_2^r$ . 5

**Assumption:** (factoring assumption for  $N$ )

Given  $N$ , it is infeasible to find  $P$  or  $Q$ .

**Theorem 1 (Factoring assumption  $\Rightarrow$  CDH assumption).**

If the factoring  $N$  is infeasible, the CDH assumption for  $QR_N$  holds.

The proof is in Section C of the appendix.

## 5 Some Technicalities

**Lemma 5.** Let  $g$  be a generator of  $QR_N$  and  $w \in {}_{\mathbb{R}} \mathbb{Z}_{pq}$ . The value  $g_2 = g^w$  is a uniformly distributed random value in  $QR_N$ . With overwhelming probability,  $g_2$  is a generator for  $QR_N$ .

*Proof.* Clearly,  $g_2$  is uniformly distributed. By Lemma 4,  $g_2$  is a generator for  $QR_N \Leftrightarrow w \in \mathbb{Z}_{pq}^*$ . Hence,  $\text{pr}[g_2 \text{ is a generator for } QR_N] = (p-1)(q-1)/pq$ .  $\square$

**Lemma 6.** If it is feasible to find any pair  $(\alpha, \beta) \in \mathbb{Z}_{pq}$  with  $(\alpha - \beta) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$ , it is feasible to factorise  $N$ .

<sup>4</sup> An alternative view would be to consider two distributions  $D_4$  and  $R_4$  of quadruples  $(g, g_2, u_1, u_2)$ . The distribution of  $g$  is the same for  $D_4$  and  $R_4$ , and  $g$  is a generator. Apart from that, we don’t specify how  $g$  is actually chosen. The values  $g_2$ ,  $u_1$  and  $u_2$  are either chosen according to  $D$ , or according to  $R$ .

<sup>5</sup> Since  $g \in QR_N$  is a generator,  $\log_g(x)$  is uniquely defined for  $x \in QR_N$ .

*Proof.* Let  $g$  be a generator for  $\text{QR}_N$ . If  $(\alpha - \beta) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$ , then  $\text{ord}(g^{\alpha-\beta}) \in \{p, q\}$  and thus, we can compute  $\text{gcd}(g^{\alpha-\beta} - 1, N) \in \{P, Q\}$ .  $\square$

**Lemma 7.** *Let  $g$  be a generator for  $\text{QR}_N$  and  $g_2 \in_{\mathbb{R}} \text{QR}_N$ . If it is feasible to choose  $u_1, u_2$  such that  $u_1 = g^{r_1}$ ,  $u_2 = g_2^{r_2}$ , and  $(r_2 - r_1) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$ , it is feasible to factorise  $N$ .*

*Proof.* Choose  $g_2$  as suggested in Lemma 5.  $w \in_{\mathbb{R}} \mathbb{Z}_{pq}$ ;  $g_2 = g^w$ . Since  $(r_2 - r_1) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$ ,  $\text{ord}(g^{r_2-r_1}) \in \{p, q\}$ . Similarly,  $\text{ord}(g_2^{r_2-r_1}) \in \{p, q\}$ , and thus  $\text{gcd}(g_2^{r_2-r_1}, N) \in \{P, Q\}$ . Due to  $g_2^{r_2-r_1} = g_2^{r_2}/g_2^{r_1} = u_2/u_1^w$ , and since we know  $w$ , we actually can compute  $g_2^{r_2-r_1}$  and thus factorise  $N$ .  $\square$

Now we describe a simulator for the Cramer-Shoup cryptosystem. Its purpose is not to be actually used for key encapsulation and decapsulation, but as a technical tool for the proof of security. If an adversary mounts an attack against the Cramer-Shoup cryptosystem, the simulator may provide the responses, instead of an “honest” Cramer-Shoup oracle. Note that the adversary can make many key decapsulation queries, but only one single key encapsulation query.

## A Simulator for the Cramer-Shoup Cryptosystem in $\text{QR}_N$

- Generate the public key:
  - Let the values  $g$ ,  $N$  and  $H$  and a triple  $(g_2, u_1, u_2) \in \text{QR}_N^3$  be given.
  - Choose  $x_1, x_2, y_1, y_2, z_1, z_2 \in_{\mathbb{R}} \mathbb{Z}_{pq}$ . Compute  $c = g^{x_1} g_2^{x_2}$ ,  $d = g^{y_1} g_2^{y_2}$ , and  $e = g^{z_1} g_2^{z_2}$ .  $\text{6}$  The public key is  $\text{PK}=(N, g, H, g_2, c, d, e)$ .
- Key Encapsulation  $\text{KE}(\text{PK})$ :
  - Compute  $k = u_1^{z_1} u_2^{z_2}$ ,  $\alpha = H(u_1, u_2)$ ,  $t = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$ .
  - The ciphertext is  $(u_1, u_2, t)$ , the encapsulated key is  $k$ .
- Key Decapsulation  $\text{KD}(\text{SK}, (U_1, U_2, T))$ :
  - Compute  $K' = U_1^{z_1} U_2^{z_2}$ ,  $A' = H(U_1, U_2)$ ,  $T' = U_1^{x_1+y_1A'} U_2^{x_2+y_2A'}$ .
  - If  $T = T'$  then output  $K'$ , else reject.

## 6 A Proof of Security in the Standard Model

In this section, we prove the security of the Cramer-Shoup Cryptosystem in  $\text{QR}_N$  in the standard model. The proof is based on three lemmas.

### Theorem 2 (Security in the standard model).

*If  $H$  is TC-resistant and both the DDH assumption for  $\text{QR}_N$  and the factoring assumption for  $N$  hold, the Cramer-Shoup cryptosystem in  $\text{QR}_N$  is ACC-secure.*

**Lemma 8.** *If the triple  $(g_2, u_1, u_2)$  given to the simulator is distributed according to distribution  $D$ , an adversary cannot statistically distinguish between the behavior of the simulator and the Cramer-Shoup cryptosystem itself.*

<sup>6</sup> In contrast to the simulator, the cryptosystem itself implicitly defines  $z_2 = 0$ .

*Proof.* If  $(g_2, u_1, u_2)$  is distributed according to  $D$ , a value  $r$  exists such that  $u_1 = g^r$  and  $u_2 = g_2^r$ . We show that the simulator's responses are statistically indistinguishable from the *real* cryptosystem's responses.

Consider the key encapsulation query. The simulator computes

$$k = u_1^{z_1} u_2^{z_2} = g^{r z_1} g_2^{r z_2} = (g^{z_1} g_2^{z_2})^r = e^r,$$

$$\alpha = H(u_1, u_2) \text{ and}$$

$$t = u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha} = g^{r x_1 + r y_1 \alpha} g_2^{r x_2 + r y_2 \alpha} = g^{r x_1} g_2^{r x_2} g^{r y_1 \alpha} g_2^{r y_2 \alpha} = c^r d^{r \alpha}.$$

The distribution of the response  $((g_2, u_1, u_2), k)$  is identical to the distribution of the cryptosystem's response.

Now consider the key decapsulation queries. If a query  $(U_1, U_2, T)$  is valid, i.e., if a value  $R \in \mathbb{Z}_{pq}$  exists with  $U_1 = g^R$  and  $U_2 = g_2^R$ , the simulator's response is the same as the response the cryptosystem provides. Both the simulator and the cryptosystem reject  $(U_1, U_2, T)$  if  $T \neq T' = U_1^{x_1 + y_1 A'} U_2^{x_2 + y_2 A'}$ , and else output  $K' = U_1^{z_1} U_2^{z_2} = (g^R)^{z_1} (g_2^R)^{z_2} = (g^{z_1} g_2^{z_2})^R = e^R$ . It remains to show that both the cryptosystem and the simulator (given  $(g_2, u_1, u_2)$  distributed according to  $D$ ) reject all invalid key decapsulation queries with overwhelming probability – and thus essentially behave identically.

The decision to reject an invalid ciphertext  $(U_1, U_2, T)$  depends on four random values  $x_1, x_2, y_1, y_2 \in \mathbb{Z}_{pq}$ . A part of the public key are the values  $c$  and  $d$  with  $c = g^{x_1} g_2^{x_2} = g^{x_1} g^{w x_2}$  and  $d = g^{y_1} g_2^{y_2} = g^{y_1} g^{w y_2}$ , i.e.,

$$l_c := \log_g(c) = x_1 + w x_2 \iff x_1 = l_c - w x_2 \text{ and} \quad (1)$$

$$l_d := \log_g(d) = y_1 + w y_2. \iff y_1 = l_d - w y_2 \quad (2)$$

These equations<sup>7</sup> provide public information about the quadruple  $(x_1, x_2, y_1, y_2)$  of secret values. The response to the encapsulation query provides another equation  $\log_g(t) = r x_1 + r y_1 \alpha + r w x_2 + r w y_2 \alpha$ , however  $\log_g(t) = r l_c + r l_d \alpha$ , i.e., this new equation linearly depends on Equations (1) and (2), and thus provides no new information about  $(x_1, x_2, y_1, y_2)$ . This still leaves  $(pq)^2$  possibilities for the quadruple  $(x_1, x_2, y_1, y_2)$ .

Assume  $g_2$  to be a generator for  $\text{QR}_N$ . (By Lemma 5, this is overwhelmingly probable.) Let the ciphertext  $(U_1, U_2, T)$  be invalid. Thus,  $R_1 \neq R_2$  exist with  $U_1 = g^{R_1}$  and  $U_2 = g_2^{R_2}$ . To answer the query, the values  $K' = U_1^{z_1} U_2^{z_2}$  (or  $K' = U_1^{z_1}$ ),  $A' = H(U_1, U_2)$ , and  $T' = U_1^{x_1 + y_1 A'} U_2^{x_2 + y_2 A'} = g^{R_1 x_1 + R_1 y_1 A'} g_2^{R_2 x_2 + R_2 y_2 A'}$  are computed, which provides the equation

$$l_{T'} := \log_g(T') = R_1 x_1 + R_1 y_1 A' + w R_2 x_2 + w R_2 y_2 A'. \quad (3)$$

Equations (1) and (2) can be used to eliminate the variables  $x_1$  and  $y_1$ :

$$\begin{aligned} l_{T'} &= R_1 l_c - R_1 w x_2 + R_1 l_d A' - R_1 w y_2 A' + w R_2 x_2 + w R_2 y_2 A' \\ &= R_1 l_c + R_1 l_d A' + w x_2 (R_2 - R_1) + w y_2 A' (R_2 - R_1) \end{aligned}$$

<sup>7</sup> It is vital that  $l_c$  and  $l_d$  are uniquely defined. We need not actually compute  $l_c$  or  $l_d$ .

By Lemma 5 and Lemma 7 we know that with overwhelming probability and under the factoring assumption both  $w$  and  $(R_2 - R_1)$  are invertible mod  $pq$ . If these two values are invertible, we may fix the value  $y_2$  arbitrarily and there always exists a uniquely defined value

$$x_2 = \frac{l_{T'} - R_1 l_c - R_1 A' l_d - w y_2 A' (R_2 - R_1)}{w(R_2 - R_1)}$$

to prevent the rejection of the invalid ciphertext  $(U_1, U_2, T)$ . Each time an invalid ciphertext is rejected, this eliminates at most  $pq$  of the  $(pq)^2$  possible quadruples  $(x_1, x_2, y_1, y_2)$ .  $\square$

**Lemma 9.** *If the triple  $(g_2, u_1, u_2) \in QR_N^3$  given to the simulator is distributed according to distribution  $R$ , the simulator rejects all invalid ciphertexts with overwhelming probability.*

*Proof.* Recall that the rejection of an invalid ciphertext  $(U_1, U_2, T)$  depends on the quadruple  $(x_1, x_2, y_1, y_2) \in QR_N$  of secret values, and that the public key provides the two linear Equations 1 and 2 to narrow down the number of possibilities for  $(x_1, x_2, y_1, y_2)$  to  $(pq)^2$ . The response to the encapsulation query provides the value  $t = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$  and thus a linear equation

$$l_t := \log_g(t) = r_1 x_1 + r_1 y_1 \alpha + w r_2 x_2 + w r_2 y_2 \alpha. \quad (4)$$

By using Equations 1 and 2, we can eliminate the variables  $x_1$  and  $y_1$ :

$$\begin{aligned} l_t &= r_1 l_c - r_1 w x_2 + r_1 l_d \alpha - r_1 w x_2 \alpha + w r_2 x_2 + w r_2 y_2 \alpha \\ &= r_1 l_c + r_1 l_d \alpha + w x_2 (r_2 - r_1) + w y_2 \alpha (r_2 - r_1) \\ \Rightarrow y_2 &= \frac{l_t - r_1 l_c - r_1 l_d \alpha - w x_2 (r_2 - r_1)}{w(r_2 - r_1) \alpha} \end{aligned}$$

An invalid ciphertext  $(U_1, U_2, T)$  is rejected, except when Equation 3 holds, which means  $T' = T$ . Recall  $\alpha = H(u_1, u_2)$  and  $A' = H(U_1, U_2)$  and consider three cases:

- Case 1,  $(U_1, U_2) = (u_1, u_2)$ : By the definition of an ACC attack, we require  $t \neq T$ , and thus the key decapsulation query  $(U_1, U_2, T)$  will be rejected.
- Case 2,  $(U_1, U_2) \neq (u_1, u_2)$  and  $\alpha = A'$ : This is a collision for  $H$  for the target  $(u_1, u_2)$ , which contradicts the assumption for  $H$  to be TC-resistant.
- Case 3:  $(U_1, U_2) \neq (u_1, u_2)$  and  $\alpha \neq A'$ : We have four unknowns  $x_1, x_2, y_1, y_2 \in QR_N$ , and four Equations 1, 2, 3, and 4 describe their relationship. By solving this system of linear equations we get

$$y_2 = \frac{l_{T'} - r_1 l_c - \frac{l_t - r_2 l_c - r_1 l_d \alpha}{r_2 - r_1} (R_2 - R_1) - A' R_1 l_d}{(R_2 - R_1) w (A' - \alpha)},$$

which uniquely determines  $y_2$  if all the four values  $(r_2 - r_1)$ ,  $(R_2 - R_1)$ ,  $w$ , and  $(A' - \alpha)$  are invertible in  $\mathbb{Z}_{pq}$ .<sup>8</sup> The invertibility of  $(r_2 - r_1)$  and

<sup>8</sup> This implies that the four linear equations 1, 2, 3, and 4 are linearly independent.

$(R_2 - R_1)$  follows from Lemma 7, the invertibility of  $w$  follows from Lemma 5, and the invertibility of  $(A' - \alpha)$  follows from Lemma 6.  $\square$

**Lemma 10.** *Let  $k$  be the encapsulated key in the response for the encapsulation query. If the triple  $(g_2, u_1, u_2) \in QR_N^3$  given to the simulator is distributed according to distribution  $R$ , it is infeasible for the adversary to distinguish between  $k$  and a uniformly distributed random value.*

*Proof.* We set  $r_1 = \log_g(u_1)$  and  $r_2 = \log_{g_2}(u_2)$ . Assume that  $g_2 = g^w$  is a generator for  $QR_N$  and that  $r_1 \neq r_2$ . Both assumptions hold with overwhelming probability. Now we prove: *If all invalid decapsulation queries are rejected during the simulation, then under the factoring assumption it is infeasible for the adversary to distinguish between  $k$  and a random value.*

Observe that  $k$  only depends on the two random values  $z_1, z_2 \in QR_N$ . Since  $e = g^{z_1} g_2^{z_2}$ , the public key provides one linear equation

$$l_e := \log_g(e) = z_1 + wz_2 \iff z_1 = l_e - wz_2. \quad (5)$$

The rejection of an invalid key decapsulation query does not depend on  $z_1$  and  $z_2$ . If the decapsulation query  $(U_1, U_2, T)$  is valid and not rejected, we have a value  $R$  such that  $U_1 = g^R$  and  $U_2 = g_2^R$ . By  $\log_g(k) = Rz_1 + Rwz_2 = R\log_g(e)$  this provides another equation, linearly depending on Equation 5. The response for the key encapsulation query consists of a ciphertext  $(u_1, u_2, t)$  and a key  $k = u_1^{z_1} u_2^{z_2} = g^{r_1 z_1} g^{w r_2 z_2}$ , which provides a linear equation

$$l_k := \log_g(k) = r_1 z_1 + w r_2 z_2 = r_1 l_e - r_1 w z_2 + r_2 w z_2 = r_1 l_e + w z_2 (r_2 - r_1), \quad (6)$$

which finally gives

$$z_2 = \frac{l_k - r_1 l_e}{w(r_2 - r_1)}.$$

As before, we argue that with overwhelming probability and under the factoring assumption both  $w$  and  $(r_2 - r_1)$  are invertible in  $\mathbb{Z}_{pq}$ . If  $w$  and  $(r_2 - r_1)$  are invertible, then a unique value  $z_2$  exists for every key  $k \in QR_N$ .  $\square$

*Proof (Theorem 2).* If the adversary can break the cryptosystem by distinguishing a *real* encapsulated key from a *random* one, she can do so as well in the simulation, if the simulator input is chosen according to distribution  $D$  (Lemma 8). Since she cannot distinguish a real key from a random key in the simulation if the simulator input is distributed according to  $R$  (Lemmas 9 and 10), being able to break the cryptosystem means being able to distinguish distribution  $D$  from distribution  $R$ , contradicting the DDH assumption for  $QR_N$ .  $\square$

*Remark 1 (Strengthening Theorem 2 by avoiding the factoring assumption).*

If  $H$  is TC-resistant and the DDH assumption for  $QR_N$  holds, the Cramer-Shoup Cryptosystem in  $QR_N$  is ACC-secure.

To verify this, assume that the adversary somehow learns the factors  $P$  and  $Q$  of  $N$ . Then the DDH-problem for  $QR_N$  is hard if and only if both the DDH problem for  $QR_P$  and the DDH problem for  $QR_Q$  are hard. But given  $P$  and  $Q$  and an oracle to mount an ACC-attack against the Cramer-Shoup Cryptosystem for  $QR_N$ , we can use this oracle to solve the DDH problem for either  $QR_P$  or  $QR_Q$ . In this case, the DDH problem for  $QR_N$  is feasible.

## 7 An Extension and Its Security

We describe how to extend the Cramer-Shoup cryptosystem, dealing with a hash function  $h$ , which may be used like a random oracle ( $\rightarrow$  Figure 1):

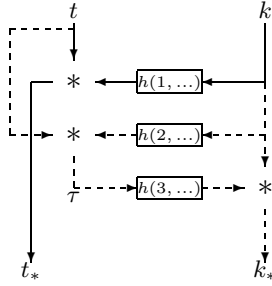


Fig. 1. The  $h$ -extension: converting  $t$  and  $k$  into  $t_*$  and  $k_*$ .

### 7.1 The Extended Scheme and Its Abstract Security

#### Cramer-Shoup Cryptosystem in $QR_N$ with $h$ -Extension:

- The key pair  $(PK, SK)$  is the same as for the non-extended Cramer-Shoup cryptosystem. Let  $h$  be a function  $h : \{1, 2, 3\} \times QR_N^3 \rightarrow QR_N$ .
- Extend key encapsulation by computing  $t_* = t * h(1, k, u_1, u_2)$  ( $\rightarrow$  solid arrows in Figure 1) and  $\tau = t * h(2, k, u_1, u_2)$  and  $k_* = k * h(3, \tau, u_1, u_2)$  ( $\rightarrow$  dashed arrows). The ciphertext is  $(u_1, u_2, t_*)$ , the encapsulated key is  $k_*$ .
- Decapsulate the ciphertext  $(U_1, U_2, T_*) \in QR_N^2 \times \mathbb{Z}_N^*$  by computing  $K'$ ,  $T'$  as before, and reject if  $T' * h(1, K', U_1, U_2) \neq T_*$ . Else compute  $\tau' = T' * h(2, K', U_1, U_2)$  and output  $K'_* = K' * h(3, \tau', U_1, U_2)$ .

#### Theorem 3 (Security of $h$ -extended scheme in standard model).

Let  $h$  be any efficient function  $h : \{1, 2, 3\} \times QR_N^3 \rightarrow QR_N$ . If  $H$  is TC-resistant and both the DDH assumption for  $QR_N$  and the factoring assumption for  $N$  hold, the Cramer-Shoup cryptosystem in  $QR_N$  is ACC-secure.



*Proof.* Observe that the simulator described in Section 5 computes the values  $k$  and  $t$  when dealing with an encapsulation query. Also, being asked to decapsulate the ciphertext  $(U_1, U_2, T)$ , the same simulator computes the values  $K'$  and  $T'$  from  $U_1$  and  $U_2$ . Thus, it is straightforward to apply the  $h$ -extension to the simulator. Since  $h$  is efficient, the extended simulator is efficient, too.

Using the extended simulator instead of the original one, the proof for Theorem 2 is applicable to Theorem 3.  $\square$

**Theorem 4 (Security of  $h$ -extended scheme in random oracle model).**

*If the function  $h$  is modeled as a random oracle, the  $h$ -extended scheme is ACC-secure under the factoring assumption.*

*Proof.* Let  $N$  and  $H$  be given. Consider an adversary with a non-negligible advantage to win the attack game. In the following experiment, we modify the key generation and we describe how to respond to the adversary's oracle queries, including queries to the random oracle. We start with the key generation:

- Choose  $\beta \in_{\mathbb{R}} \{1, \dots, \lfloor N/4 \rfloor - 1\}$ ,  $\alpha \in_{\mathbb{R}} \mathbb{Z}_N^*$  and compute  $e := \alpha^2$ .
- Choose  $u_1 \in_{\mathbb{R}} \text{QR}_N$  and compute  $g := u_1^{2\beta}$ . (We will search for  $k = e^{\log_g(u_1)}$ , i.e., for the value  $k$  with  $k^{2\beta} = e$ . If we find  $k$ , we have a 50% chance that  $\gcd(k^\beta - \alpha, N) \in \{P, Q\}$  holds, providing us with the factorisation of  $N$ .)
- Choose  $w \in_{\mathbb{R}} \mathbb{Z}_{pq}$  and compute  $g_2 = g^w$  and  $u_2 = u_1^w$ .
- Choose  $x_1, x_2, y_1, y_2 \in_{\mathbb{R}} \mathbb{Z}_{pq}$  and compute  $c = g^{x_1} g_2^{x_2}$  and  $d = g^{y_1} g_2^{y_2}$ .
- Use  $(N, g, H, g_2, c, d, e)$  as the public key.

The response to the key encapsulation query is the ciphertext  $(u_1, u_2, t_*)$  and the encapsulated key  $k_*$  with  $t_*, k_* \in_{\mathbb{R}} \text{QR}_N$ .

Let  $(U_1, U_2, T_*)$  be a key decapsulation query. We respond as follows:

- Compute  $T' = U_1^{x_1+y_1A'} U_2^{x_2+y_2A'}$ .
- Consider values  $K'$  with queries for  $\delta_1 = h(1, K', U_1, U_2)$  to the random oracle. Verify, if for one such value  $K'$  the equation

$$(K')^{2\beta} = U_1 \tag{7}$$

holds. If not, or if  $T_* \neq T' * \delta_1$ , then reject.

- Else ask the random oracle for  $\delta_2 = h(2, K', U_1, U_2)$ , compute  $\tau = T' * \delta_2$ , ask for  $\delta_3 = h(3, \tau, U_1, U_2)$ , and respond  $K'_* = K' * \delta_3$  to the adversary.

A random oracle query to compute  $h(I, X, U_1, U_2)$  (with  $I \in \{1, 2, 3\}$  and  $X, U_1, U_2 \in \text{QR}_N$ ) may be asked either by the adversary, or by ourselves when answering a key decapsulation query. The answer is computed as follows:

1. If we have been asked for  $h(I, X, U_1, U_2)$  before, repeat the same answer.
2. Else, if  $I \in \{1, 2\}$ ,  $u_1 = U_1$ ,  $u_2 = U_2$ , and  $X^{2\beta} = e$ , print  $X$  and abort.
3. Else choose  $Y \in_{\mathbb{R}} \text{QR}_N$  and respond  $Y$ .

Observe that if we never abort ( $\rightarrow$  Step [2](#)), the adversary cannot distinguish  $h$  from a random function over the same domain. On the other hand, assume that we abort the experiment, having found a value  $X$  with  $X^{2\beta} = e$ , i.e., a square root (mod  $N$ ) of  $e$ . Initially, we know two square roots of  $e$ , namely  $\pm\alpha$ . Since the adversary has no information about  $\alpha$ , except for  $e = \alpha^2$ ,  $X^\beta \neq \pm\alpha$  holds with probability  $1/2$ . In this case, we can factorise  $N$  by computing  $\gcd(X^\beta - \alpha, N) \in \{P, Q\}$ . This shows: *If  $\pi$  is the probability to abort the experiment, we can factorise  $N$  with the probability  $\pi/2$  after running the experiment once.*

Now, we deal with three different games:

1. The attack game with the “real” encapsulated key  $k_*$ ,
2. the attack game where  $k_*$  is replaced by a random value, and
3. the experiment we defined for the current proof.

As it turns out, the adversary cannot distinguish the experiment from either of the attack games, except when we abort the experiment:

- The public key values  $g, g_2, c, d$ , and  $e$  are independent uniformly distributed random values in  $\text{QR}_N$  – in the attack games, as in the experiment.
- In the attack games, the values  $u_1$  and  $u_2$  from the encapsulation query satisfy the equation  $u_2 = g_2^{\log_g(u_1)}$ , with  $u_1 \in_{\text{R}} \text{QR}_N$ . For one of the attack games, the values  $t_*$  and  $k_*$  depend on  $t$  and  $k$  (and  $h$ ), while for the other one,  $t_*$  depends on  $t$  and  $k$ , while  $k_*$  is chosen at random.

In the experiment,  $u_1 \in_{\text{R}} \text{QR}_N$  and  $u_2 = g_2^{\log_g(u_1)}$  as well. The value  $t_*$  cannot be distinguished from a uniformly distributed random value without asking for  $h(1, k, u_1, u_2)$  (and then aborting). The value  $k_*$  cannot be distinguished without asking for  $h(3, \tau, u_1, u_2)$ . Asking this query is infeasible without having asked for  $\delta_2 = h(2, k, u_1, u_2)$  (followed by an abortion), since  $\tau$  depends on  $\delta_2$ .

- Consider a decapsulation query  $(U_1, U_2, T_*)$ . Let  $K'$  be defined by Equation [7](#). If  $h$  is a well-defined function, there is a unique well defined value  $T'_*$  such that a ciphertext  $(U_1, U_2, T'_*)$  has to be accepted, and every ciphertext  $(U_1, U_2, T_*)$  with  $T_* \neq T'_*$  has to be rejected. Without asking for  $h(1, K', U_1, U_2)$ , the adversary cannot predict  $T'_*$ , and any ciphertext  $(U_1, U_2, T_*)$  chosen by the adversary is rejected with overwhelming probability in the attack games and with probability 1 in the experiment.

If the adversary had asked for  $h(1, K', U_1, U_2)$ , the computation of  $T'_*$  and  $K'_*$  is exactly the same in the experiment as in the attack games.  $\square$

## 7.2 The Concrete Security of the Extended Scheme

Note that the reduction in the proof of Theorem [4](#) is very efficient. We quantify this by describing the *concrete security* against a generic adversary, i.e., against an adversary who treats the hash function  $h$  like a random oracle.

**Theorem 5 (Concrete security of  $h$ -extended scheme in r. o. model).**

*Let  $\mathcal{A}$  be a generic ACC adversary, allowed to ask one key encapsulation query,*

$q_{\text{KD}}$  key decapsulation queries, and  $q_1 + q_2 + q_3$  random oracle queries, namely  $q_i$  random oracle queries of the form  $h(i, \dots)$ . Assume  $\mathcal{A}$  takes the running time  $T_{\mathcal{A}}$  and achieves the advantage  $a_{\mathcal{A}}$  when distinguishing between the attack game with the “real” and the attack game with a random encapsulated key.

Then an algorithm  $\mathcal{F}$  exists to find the factors  $P$  and  $Q$  of  $N$  with at least the probability  $a_{\mathcal{A}}/2 - (q_3 + 2q_{\text{KD}})/pq$ . The expected running time for  $\mathcal{F}$  is at most  $T_{\mathcal{A}} + T_{\delta}$ , with  $T_{\delta}$  being linear in the total number  $q_{\Sigma} = (1 + q_{\text{KD}} + q_1 + q_2 + q_3)$  of oracle queries. More specifically,  $T_{\delta}$  is the time for doing  $7 + 3q_{\text{KD}} + q_1 + q_2$  exponentiations mod  $N$  and  $O(q_{\Sigma})$  other operations.

*Proof.* The proof of Theorem 4 already describes what we call algorithm  $\mathcal{F}$ , here: Run the key generation and then invoke the distinguishing adversary  $\mathcal{A}$ , providing all responses to  $\mathcal{A}$ ’s oracle queries. To prove Theorem 5 we concretely analyse running time and probability of success of this algorithm.

Running time:

During key generation, we compute seven values by exponentiation mod  $N$ :  $u_1^{2\beta}$ ,  $g^w$ ,  $u_1^w$ ,  $g^{x_1}$ ,  $g_2^{x_2}$ ,  $g^{x_1}$ ,  $g_2^{y_2}$ . When responding to the key encapsulation query, no exponentiations are necessary. Responding to a random oracle query  $h(1, \dots)$  or  $h(2, \dots)$  may require to compute  $X^{2\beta}$ . Queries  $h(3, \dots)$  can be answered without computing any exponentiations.

Key decapsulation queries are slightly more complicated and may need up to three exponentiations. Two are needed to compute  $T'$ . The values  $(K')^{2\beta}$  have already been computed when dealing with a random oracle query  $h(1, \dots)$  and may have been stored in a table. But responding to a key decapsulation query may require to make two additional calls  $h(2, \dots)$  and  $h(3, \dots)$  to the random oracle, and calling  $h(2, \dots)$  may require another (third) exponentiation.

Thus the total number of exponentiations mod  $N$  is at most

$$3q_{\text{KD}} + q_1 + q_2 + 7.$$

Similarly, we can count the total number of other operations, which is no more than linear in  $q_{\Sigma}$ , as well.

Note that the random oracle may have to respond to at most  $q_1$  queries  $h(1, \dots)$ , but to at most  $q_{\text{KD}} + q_2$  queries  $h(2, \dots)$  and  $q_{\text{KD}} + q_3$  queries  $h(3, \dots)$ . The reason is, that computing the answer to a decapsulation query may include two additional random oracle queries  $h(2, \dots)$  and  $h(3, \dots)$ .

Probability:

$\mathcal{A}$  cannot distinguish between the two attack games without asking for  $h(3, \tau, u_1, u_2)$ , where  $\tau = h(2, k, u_1, u_2) * t$ . If  $\mathcal{A}$  ever asks for  $h(2, k, u_1, u_2)$ , the simulator aborts and  $\mathcal{F}$  will factorise  $N$  with a 50 % probability of success. Else,  $\mathcal{A}$  has no (Shannon-) information about  $\tau$ . In this case, and since at most  $q_3 + q_{\text{KD}}$  queries of the form  $h(3, \dots)$  are to be answered, the probability that any of these is of the form  $h(3, \tau, u_1, u_2)$  is at most  $(q_3 + q_{\text{KD}})/pq$ .

When might the adversary be able to distinguish either of the attack games from the experiment we define in the proof of Theorem 4, i.e., from the behavior

of algorithm  $\mathcal{F}$ ? The experiment behaves exactly like any of the attack games, with the following two exceptions:

1.  $\mathcal{A}$  asks for  $h(I, X, U_1, U_2)$  with  $I \in \{1, 2\}$  and  $X^{2\beta} = e$ . In this case, the experiment is aborted (and  $\mathcal{F}$  has a 50 % chance of factorising  $N$ ).
2.  $\mathcal{A}$  asks for the decapsulation of a ciphertext  $(U_1, U_2, T_*)$ , without having asked for  $h(1, K', U_1, U_2)$  before ( $K'$  is defined in Equation (7)). In this case,  $\mathcal{F}$  always rejects, while the attack games reject with the probability  $1/pq$ .

Since  $\mathcal{A}$  can ask for the decapsulation of  $q_{\text{KD}}$  ciphertext, the entire probability that any random ciphertext is not rejected in an attack game is  $\leq q_{\text{KD}}/pq$ .

Even if  $\mathcal{A}$  could always distinguish the “real” encapsulated key from a random value when asking for  $h(3, \tau, u_1, u_2)$  without asking for  $h(2, k, u_1, u_2)$  before, or when a random ciphertext  $(U_1, U_2, T_*)$  in a key decapsulation query is not rejected, the probability for  $\mathcal{F}$  to factorise  $N$  would not be less than

$$\frac{a_{\mathcal{A}}}{2} - \frac{q_3 + q_{\text{KD}}}{pq} - \frac{q_{\text{KD}}}{pq}.$$

□

*Remark 2 (Practical consequences of Theorem 5).*

Theorem 5 counts modular exponentiations and mentions “other operations”. These are simpler arithmetic operations (e.g. multiplication mod  $N$ ), choosing random values ( $\in_{\mathbb{R}} \text{QR}_N$ ,  $\in_{\mathbb{R}} \mathbb{Z}_N^*$ , and  $\in_{\mathbb{R}} \mathbb{Z}_{pq}$ ), and hash table look-up and update operations. In practice, none of these operations should be slower than an exponentiation mod  $N$ . Thus, the running time for algorithm  $\mathcal{F}$  is  $T_{\mathcal{A}} + O(q_{\Sigma} * T_N)$ , where  $T_N$  is the time for computing an exponentiation mod  $N$ .

For any reasonable choice of  $N$ , the probability that  $F$  actually factorises  $N$  is extremely close to  $a_{\mathcal{A}}/2$ .

### 7.3 Comparison to Shoup’s Technique

The approach in this section has been inspired by Shoup [13], who also described a “hedged” variant of the Cramer-Shoup Scheme, being both

- provably secure in the standard model under a “strong” assumption *and*
- provably secure in the random oracle model under a “weak” assumption.

In [13], the “strong” assumption is the DDH assumption for a group  $G$  of prime order. The “weak” assumption is the CDH assumption for  $G$ . As was stressed in [6] (see also [13, Remark 4]), the reduction in the random oracle model is quite inefficient, since it is relative to a DDH oracle:

- Let the DDH assumption for  $G$  be false. I.e., a polynomial-time algorithm  $A_1$  with a *significant* DDH advantage exists. By standard amplification techniques (calling  $A_1$  polynomially often), we get  $A_2$ , which achieves an overwhelming DDH advantage. Note that the DDH-oracle  $A_2$  is “efficient” in the sense of Complexity Theory, but may be quite inefficient in practice.

Assume an efficient generic ACC adversary  $A$  exists to break the hedged Cramer-Shoup variant [13]. The reduction in [13] describes how to use the adversary  $A$  as a tool to break the CDH assumption for  $G$ . The reduction requires to call  $A_2$  each time when  $A$  asks a new random oracle query.

- Consider a hypothetical example (using viciously chosen numbers):

*Let, for some choice of  $G$ ,  $A_1$  run in  $2^{30}$  computer clocks. Thus,  $A_1$  could qualify as “practically efficient”. If  $A_2$  executes  $A_1$   $2^{30}$  times,  $A_2$  could be considered “hard, but feasible” on a massively parallel computer. Now consider an efficient generic ACC adversary  $A$  making  $2^{30}$  random oracle queries. The reduction provides an algorithm to solve the CDH problem for  $G$ , but this algorithm would require more than  $2^{90}$  units of time.*

Thus, an efficient generic ACC attack against the scheme does not necessarily reduce to a practical solution for the CDH problem for  $G$ .

As explained above, the reduction in the current paper is quite efficient, using linearly many *moderately simple* operations (such as exponentiations mod  $N$ ), but no *potentially complex* operation (such as the DDH oracle in [13]).

Also note that we do not assume the hash function  $H$  to be TC-resistant, for Theorem 4 in contrast to [13, Theorem 3].

On the other hand, the random oracle security in the current paper is based on the factoring assumption, not on the CDH assumption. This may be seen as a disadvantage. By generalising the technique from [13] for  $\text{QR}_N$ , we might be able to use the CDH assumption for  $\text{QR}_N$  instead, which is at least as strong as the factoring assumption for  $N$ , see Theorem 11.

A rather technical difference to our approach is that [13] introduces the notion of a *pair-wise independent hash function* (PIH) and combines a PIH with a random oracle. The PIH is required for the security result in the standard model (i.e., for the counterpart of Theorem 3 in the current paper).

## 8 Final Remarks and Discussion

*Remark 3 (The input for  $KD$ ).*

Note that the input  $(U_1, U_2, T^*) \in \text{QR}_N^2 \times \mathbb{Z}_N^*$  is under control of the adversary. If  $x$  is a number, it is easy to verify whether  $x \in \mathbb{Z}_N^*$ , but it may be difficult to verify  $x \in \text{QR}_N$ . We can deal with this problem by using  $\text{KE}'$  and  $\text{KD}'$  instead of  $\text{KE}$  and  $\text{KD}$ :

- $\text{KE}'$ : Compute  $\text{KE}$  and replace  $k$  by  $k^2$  and  $t$  by  $t^2$ .
- $\text{KD}'(\text{SK}, (U_1, U_2, T))$  for  $(U_1, U_2, T) \in (\mathbb{Z}_N^*)^3$ : Compute  $\text{KD}(\text{SK}, (U_1^2, U_2^2, T))$ .

Note that  $(\text{Gen}, \text{KE}', \text{KD}')$  is as sound as  $(\text{Gen}, \text{KE}, \text{KD})$ . But for  $(U_1, U_2, T) \in (\mathbb{Z}_N^*)^3$ , the input for  $KD$  is now in  $\text{QR}_N^2 \times \mathbb{Z}_N^*$ , as it should. A similar technique can be used for the  $h$ -extension.

*Remark 4 (The hash function  $H$ ).*

Theoretically we don't need an *additional* assumption for the TC-resistance of

*H.* Based on the factoring assumption, provably secure TC-resistant (and even stronger) hash functions are known. In practice, we may prefer to use a dedicated hash function such as SHA-1 or RIPE-MD 160.

Recall that we deal with a cryptosystem which has computations in  $\text{QR}_N$ , but nobody knows (or needs to know) the order of  $\text{QR}_N$ . (Note that knowing the order of  $\text{QR}_N$  is equivalent to knowing the factors of  $N$ .)

This may be interesting for the construction of advanced cryptographic protocols, where some party knows the factorisation of  $N$  *in addition* to the secret key, while another party only knows the secret key itself. E.g., consider a variant of our scheme, where the factors of  $N$  (possibly more than just two, in contrast to the current paper) are small enough that computing the discrete log modulo any of the factors is feasible. Everyone knowing the factorisation of  $N$  can thus compute discrete logarithms mod  $N$ , and the factorisation of  $N$  may serve as a “master key”: Knowing it allows to compute the secret key from a given public key defined over the group  $\text{QR}_N$ . This approach is roughly related to key insulation [8], where “ordinary” public keys may be stored and used in a risky environment, while the master key is well protected.

From a practical point of view, it may not appear too useful to hide the order of the group from the owner of the secret key (except in the context of the advanced protocols mentioned above). In practice, the owner of the secret key might want to use the knowledge of the factors  $P$  and  $Q$  of  $N$  to improve the efficiency of key decapsulation by applying the Chinese Remaindering Theorem.

The main practical selling point for the current scheme is the improved security assurance in the random oracle model, compared to [13].

An interesting open problem is the following: *Is this paper’s hedging technique (cf. Figure 1) applicable to other cryptosystems, e.g., to the variants of the Cramer-Shoup Cryptosystem described in [7]?*

## Acknowledgement

Eike Kiltz [10] first observed Remark 1. He and the Asiacrypt referees provided many hints to improve this presentation. Ronald Cramer [4] pointed out that the framework from [7] can be used for an alternative proof of Theorem 2 and Remark 1. Following a referee’s advice to give more attention to concrete security, Theorem 5 has been added to this final version of the paper.

## References

1. M. Abdalla, M. Bellare, P. Rogaway: “DHAES: an encryption scheme based on the Diffie-Hellman problem”, preprint, March 17, 1999, <http://eprint.iacr.org/1999/007/>.
2. M. Bellare, P. Rogaway: “Random oracles are practical: a paradigm for designing efficient protocols”, ACM Computer and Communication Security ’93, ACM Press.
3. M. Bellare, A. Desai, D. Pointcheval, P. Rogaway: “Relations among notions of security for public-key encryption scheme”, Crypto ’98, Springer LNCS 1462.

4. R. Cramer, personal communication.
5. R. Cramer V. Shoup: “A practical cryptosystem, provably secure against chosen ciphertext attacks”, Crypto ’98, Springer LNCS 1462.
6. R. Cramer V. Shoup: “Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack”, revised and extended version of [5], December 17, 2001, <http://eprint.iacr.org/2001/108/>.
7. R. Cramer V. Shoup: “Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption”, preprint, 2nd version, December 12, 2001, <http://eprint.iacr.org/2001/085/>. (Extended abstract at Eurocrypt 2002.)
8. Y. Dodis, J. Katz, S. Xu, M. Yung: “Key-Insulated Public Key Cryptosystems”, Eurocrypt 2002.
9. D. Dolev, C. Dwork, M. Naor: “Non-malleable cryptography”, SIAM Journal of Computing, 2000. (Extended abstract at STOC ’91, ACM Press.)
10. E. Kiltz, personal communication.
11. M. Naor, M. Yung: “Public-key cryptosystems provably secure against chosen ciphertext attacks”, STOC ’90, ACM Press.
12. C. Rackoff, D. Simon: “Non-interactive zero knowledge proof of knowledge and chosen ciphertext attacks”, Crypto ’91, Springer LNCS.
13. V. Shoup: “Using hash functions as a hedge against chosen ciphertext attack”, Eurocrypt ’00, Springer LNCS.

## Appendix

### A Properties of the Set $QR_N$ – Proofs

In this section, we prove the Lemmas stated in Section 2. Consider the sets

$$\begin{aligned} QR_P &= \{x \in \mathbb{Z}_P^* \mid \exists a \in \mathbb{Z}_P^* : a^2 \equiv x \pmod{P}\}, \\ QR_Q &= \{x \in \mathbb{Z}_Q^* \mid \exists a \in \mathbb{Z}_Q^* : a^2 \equiv x \pmod{Q}\}, \text{ and} \\ QR_N &= \{x \in \mathbb{Z}_N^* \mid \exists a \in \mathbb{Z}_N^* : a^2 \equiv x \pmod{N}\} \end{aligned}$$

of Quadratic Residues modulo  $P$ ,  $Q$  and  $N$ . Recall the following facts (which we don’t prove in the current paper):

**Fact 1.** *The sets  $QR_N$ ,  $QR_P$ , and  $QR_Q$  are multiplicative groups.*

**Fact 2.**  *$|QR_N| = pq$ ,  $|QR_P| = p$ , and  $|QR_Q| = q$ .*

**Fact 3.** *Groups of prime order are cyclic.*

**Lemma 1.**  *$QR_N$  has a nontrivial subgroup of order  $p$  and a nontrivial subgroup of order  $q$ . Both subgroups are cyclic.*

*Proof.* Note that  $x \in \mathbb{Z}_N$  is in  $QR_N$ , if and only if  $x$  is both a Quadratic Residue mod  $P$  and a Quadratic Residue mod  $Q$ .

If  $a \equiv 1 \pmod{P}$  and  $b \equiv 1 \pmod{P}$ , then  $ab \equiv 1 \pmod{P}$ , and if both  $a$  and  $b$  are Quadratic Residues mod  $Q$ , then  $ab$  is a Quadratic Residue mod  $Q$  as well. Thus, the set

$$\{x \in \mathbb{Z}_N^* \mid \exists a \in \text{QR}_Q : x \equiv a \pmod{Q} \text{ and } x \equiv 1 \pmod{P}\}$$

is a subgroup of  $\text{QR}_N$  of the order  $|\text{QR}_Q| = q$ . Similarly, a subgroup of  $\text{QR}_N$  of the order  $p$  exists. Groups of prime order are cyclic.  $\square$

**Lemma 2.**  *$\text{QR}_N$  is cyclic. It consists of one element of the order 1,  $(p-1)$  elements of the order  $p$ ,  $(q-1)$  elements of the order  $q$ , and  $(p-1)(q-1)$  elements of the order  $pq$ .*

*Proof.* Consider  $a, b \in \text{QR}_N$  with  $\text{ord}(a) = p$ ,  $\text{ord}(b) = q$ . Due to Lemma 1, such elements  $a$  and  $b$  exist;  $\text{ord}(ab) = \text{lcm}(p, q) = pq$ , thus  $g = ab$  generates  $\text{QR}_N$ .

Due to  $\text{ord}(g) = pq$  we have  $\text{ord}(g^0) = \text{ord}(g^{ab}) = \text{ord}(1) = 1$ ,  $\text{ord}(g^i) = pq \Leftrightarrow (i > 1 \text{ and } \gcd(i, pq) = 1)$ ,  $\text{ord}(g^{kp}) = q$  for  $k \in \{1, \dots, q-1\}$ , and  $\text{ord}(g^{lp}) = q$  for  $l \in \{1, \dots, p-1\}$ .  $\square$

**Lemma 3.** *For every  $x \in \text{QR}_N$ :  $\text{ord}(x) \in \{p, q\} \Rightarrow \gcd(x-1, N) \in \{P, Q\}$ .*

*Proof.* From Lemma 2 and the proof of Lemma 1:  $\text{ord}(x) = q \Leftrightarrow X \equiv 1 \pmod{P} \Rightarrow \gcd(x-1, N) = P$ . Similarly:  $\text{ord}(x) = p \Rightarrow \gcd(x-1, N) = Q$ .  $\square$

Lemma 3 implies that an adversary who is able to find any  $x \in \text{QR}_N$  with  $\text{ord}(x) \notin \{1, pq\}$ , can factorise  $N$ . Further, if  $\text{ord}(x) = pq$ , then  $\gcd(x-1, N) = 1$ . An implication of Lemma 2 is that it is easy to find a random generator for  $\text{QR}_N$ . Choose  $x \in_{\mathbb{R}} \mathbb{Z}_N^*$  and compute  $g = x^2 \pmod{N}$ . If  $p$  and  $q$  are large,  $g$  is a generator for  $\text{QR}_N$  with overwhelming probability. In any case,  $g$  is a generator if and only if  $\text{ord}(g) \notin \{1, p, q\}$ ;  $\text{ord}(g) = 1 \Leftrightarrow g = 1$ , and Lemma 3 provides a way to check for  $\text{ord}(g) \notin \{p, q\}$ .

**Lemma 4.** *Let  $g$  be a generator for  $\text{QR}_N$ . For every  $x \in \mathbb{Z}_{pq}$ :  $\text{ord}(g^x) \in \{p, q\} \Leftrightarrow \gcd(x, pq) \in \{p, q\}$ .*

*Proof.* If  $x \equiv p \pmod{pq}$ , then  $g^{qx} = 1$  and thus  $\text{ord}(g^x) = q$ . If  $\text{ord}(g^x) = q$ , then  $(g^x)^p = 1 \Rightarrow xp \equiv 0 \pmod{pq} \Rightarrow x \equiv p \pmod{pq}$ . Thus,  $x \equiv p \pmod{pq} \Leftrightarrow \text{ord}(g^x) = q$ . Similarly, we get  $x \equiv q \pmod{pq} \Leftrightarrow \text{ord}(g^x) = p$ .  $\square$

## B ACC-Security and Lunchtime-Security

Key decapsulation queries correspond to chosen ciphertext decryption queries in the public-key (PK) world. The key encapsulation query corresponds to the PK encryption query. Here, a plaintext is chosen by the adversary, the oracle either really encrypts that plaintext or it encrypts a random plaintext, and the adversary has to distinguish between real and random. Lunchtime (i.e. non-adaptive) security deals with *all* decryption queries *before* the encryption query. ACC attacks against PK cryptosystems deal with two phases of chosen ciphertext queries, the first *before* the encryption query, the second *after* the encryption



query. (As mentioned in Footnote [1](#), some authors denote lunchtime security by “IND-CCA1” and ACC security by “IND-CCA2”. Here “IND” means “indistinguishable”. This notation has been introduced in [3](#).) In the second phase, one may not ask for the decryption of the result of the encryption query.

A definition for a lunchtime-secure KEM would require a minor modification of our definition for an ACC-secure KEM by asking the decapsulation queries *before* the encapsulation query. And a two-phase attack against a KEM with some decapsulation queries before and some after the encapsulation query – similar to the ACC attack against PK cryptosystems – can easily be simulated by our (one-phase) ACC attack.

## C The Proof for Factoring Assumption $\Rightarrow$ CDH Assumption

*Proof (Theorem [7](#)).* We describe an algorithm using a CDH oracle for  $\text{QR}_N$  as a tool to factorise  $N$ . For random inputs, the oracle succeeds with probability  $\pi$ .

- Choose  $\beta \in_{\text{r}} \mathbb{Z}_{pq}$ ,  $\alpha \in_{\text{r}} \mathbb{Z}_N^*$  and compute  $g_2 = \alpha^2$ .
- Choose  $u_1 \in_{\text{r}} \text{QR}_N$  and compute  $g = u_1^{2\beta}$ .
- Use the CDH oracle to compute  $u_2$  with  $u_2^{2\beta} = g_2$ .
- If  $u_2^\beta \not\equiv \pm\alpha \pmod{N}$ , print  $\text{gcd}(u_2^\beta - \alpha, N)$ .

Since  $\beta \in \mathbb{Z}_{pq}$  is a uniformly distributed random value (or statistically indistinguishable from uniform) so are the values  $g, g_2, u_2 \in \text{QR}_N$ . With the probability  $\pi$ , we get a random square root  $u_2^\beta$  of  $g_2$ . Two of the four square roots of  $g_2$ , namely  $\pm\alpha$  are not useful, but if  $\alpha \not\equiv \pm u_2^\beta \pmod{N}$ , then  $\text{gcd}(u_2^\beta - \alpha, N) \in \{P, Q\}$  factorises  $N$ .  $\square$

# Looking beyond XTR

Wieb Bosma<sup>1</sup>, James Hutton<sup>2</sup>, and Eric R. Verheul<sup>3</sup>

<sup>1</sup> Mathematisch Instituut, Universiteit Nijmegen,  
Postbus 9010, 6500 GL Nijmegen, The Netherlands,  
`bosma@sci.kun.nl`

<sup>2</sup> Thales e-Security,

149 Preston Road, Brighton, BN1 6BN, U.K.,  
`jamie.hutton@thales-esecurity.com`

<sup>3</sup> PricewaterhouseCoopers, GRMS Crypto Group,  
P.O. Box 85096, 3508 AB Utrecht, The Netherlands,  
`eric.verheul@nl.pwcglobal.com,pobox.com`

**Abstract.** XTR is a general method that can be applied to discrete logarithm based cryptosystems in extension fields of degree six, providing a compact representation of the elements involved. In this paper we present a precise formulation of the Brouwer-Pellikaan-Verheul conjecture, originally posed in [4], concerning the size of XTR-like representations of elements in extension fields of arbitrary degree. If true this conjecture would provide even more compact representations of elements than XTR in extension fields of degree thirty. We test the conjecture by experiment, showing that in fact it is unlikely that such a compact representation of elements can be achieved in extension fields of degree thirty.

## 1 Introduction

Many public key cryptosystems are based on the assumed intractability of the Discrete Logarithm (DL) problem: given a cyclic group  $G = \langle g \rangle$  and  $h \in G$  find  $0 \leq x < \#G$  such that  $h = g^x$ .

Any cryptosystem based on the DL problem requires a large cyclic group  $G$  as a parameter of the system. We require that exponentiation is efficient in  $G$  but that the DL problem is believed to be hard.

The seminal example of DL-based cryptosystems is Diffie-Hellman key exchange (see [6]), a method that enables two parties (Alice and Bob) to establish a shared secret key by exchanging messages over an open channel. Alice generates a random key  $2 \leq a < \#G$  and sends  $A = g^a$  to Bob. Similarly Bob generates  $2 \leq b < \#G$  and sends  $B = g^b$  to Alice. Alice and Bob can now both determine the common secret key  $S = A^b = B^a = g^{ab}$ .

The basic and original version of Diffie-Hellman key exchange uses  $G = \mathbb{F}_p^*$  where the prime  $p$  and a generator  $g$  of  $G$  are public parameters. There are other choices for the group  $G$ . For example Claus Schnorr proposed using a prime order subgroup of  $\mathbb{F}_p^*$  (see [21]). Alternatively one can use the group of points on certain elliptic curves.

In this paper we explore another choice: a carefully chosen subgroup  $G$  of prime order  $q$  of the multiplicative group of an extension field  $\mathbb{F}_{p^k}$ . We can represent elements of  $G$  by their minimal polynomials over a subfield of  $\mathbb{F}_{p^k}$  and thereby for certain values of  $k$  achieve a comparatively compact representation of the group elements involved. This is the idea behind LUC ( $k = 2$ ) and XTR ( $k = 6$ ); see Section 2.

In Section 3 we refer to a conjecture implicitly posed by Brouwer, Pellikaan and Verheul in [4] (the ‘BPV’ conjecture) concerning the size of minimal polynomial representations of elements in field extensions of arbitrary degree. In Sections 4 and 5 we prove some general results concerning the coefficients of minimal polynomials and develop precise formulations of the BPV conjecture.

Our main objective was to investigate the possibility of obtaining a more compact representation of elements than XTR for values of  $k$  larger than 6. We used a MAGMA program (described in Section 6) to conduct our investigations. Since the BPV conjecture (if true) would provide a more compact representation than XTR in field extensions of degree thirty we considered this to be the most interesting case. However we also investigated intermediate values of  $k$  and discovered (rather to our surprise) some cases that support the conjecture, although these cases do not provide a more compact representation than XTR.

In Section 7 we present the experimental results of our investigations. We show that if the conjectured relations exist in the degree 30 case then they are most likely too complicated to be of practical value.

## 2 Representing Elements by Their Minimal Polynomials

A standard method for representing elements of an extension field  $\mathbb{F}_{p^k}$  is as vectors over a subfield  $\mathbb{F}_{p^d}$ . The usual way of achieving such a representation is to use the fact that  $\mathbb{F}_{p^k} \cong \mathbb{F}_{p^d}[X]/P(X)$  where  $P$  is an irreducible polynomial of degree  $k/d$  over  $\mathbb{F}_{p^d}$ . Elements of  $\mathbb{F}_{p^k}$  are represented by residue classes modulo  $P$  and these classes can in turn be represented by the polynomials over  $\mathbb{F}_{p^d}$  of degree less than  $k/d$ . The coefficients of these polynomials enable us to express the field elements as vectors over  $\mathbb{F}_{p^d}$  of length  $k/d$ ; therefore we generally require  $k \log p$  bits to represent an element.

A well-known alternative method is to represent  $\alpha \in \mathbb{F}_{p^k}$  by its minimal polynomial over a subfield  $\mathbb{F}_{p^d}$ . This is the unique monic irreducible polynomial  $F$  over  $\mathbb{F}_{p^d}$  such that  $F(\alpha) = 0$ . We always have  $\deg(F) \leq k/d$ .

Note that when  $\deg(F) = k/d$  and  $d < k$  then the  $k/d$  non-trivial coefficients of the minimal polynomial do not determine the element uniquely: if  $\alpha_0 \in \mathbb{F}_{p^k}$  is a root of  $F$  then so are  $\alpha_1 = \alpha_0^{p^d}$ ,  $\alpha_2 = \alpha_0^{p^{2d}}$ , ...,  $\alpha_{k/d-1} = \alpha_0^{p^{k-d}}$ . The  $\alpha_i$  are the conjugates of  $\alpha_0$  over  $\mathbb{F}_{p^d}$ , and these elements are all represented by the same minimal polynomial over  $\mathbb{F}_{p^d}$ .

The minimal polynomial over  $\mathbb{F}_{p^d}$  of an element of  $\mathbb{F}_{p^k}$  will have degree  $k/d$  unless the element is contained in a subfield  $\mathbb{F}_{p^e}$  where  $\mathbb{F}_{p^d} \subseteq \mathbb{F}_{p^e} \subset \mathbb{F}_{p^k}$ . Thus at first sight it appears that for most elements of  $\mathbb{F}_{p^k}$  we would require  $k \log p$  bits to specify the  $k/d$  non-trivial coefficients of the minimal polynomial over  $\mathbb{F}_{p^d}$ ,

and that therefore we need the same number of bits to represent elements as in the representation using residue classes discussed above. However in certain cases there exist relationships between the coefficients of the minimal polynomials that enable us to reduce the number of coefficients that are required and thereby make the representation more compact. This idea is used in both LUC and XTR; we describe these methods in the two examples at the end of this section.

We now introduce the subgroups of field extensions in which we work.

**Definition 1.** In a field  $\mathbb{F}_{p^k}$  we call a subgroup of prime order  $q$  with  $q \mid \Phi_k(p)$  and  $q \nmid k$  a *cyclotomic subgroup* and denote it by  $G_{q,p,k}$ . (Here  $\Phi_k(p)$  denotes the  $k$ -th cyclotomic polynomial evaluated in  $p$ , see [8] and [15].)

We call the group of all elements of order dividing  $\Phi_k(p)$  the  $(p, k)$ -cyclotomic group and denote it by  $G_{p,k}$ .

The original Diffie-Hellman protocol uses the  $(p, 1)$ -cyclotomic group  $G_{p,1}$ , while Schnorr's variant is based in a cyclotomic subgroup  $G_{q,p,1}$ . LUC uses a cyclotomic subgroup  $G_{q,p,2}$  and XTR uses  $G_{q,p,6}$ , as we next explain.

*Example 1.* The LUC system uses minimal polynomials to represent elements of a cyclotomic subgroup  $G_{q,p,2}$  of  $\mathbb{F}_{p^2}^*$ . The minimal polynomial over  $\mathbb{F}_p$  of an element  $h \in G_{q,p,2} \setminus \{1\}$  is

$$P_h = (X - h)(X - h^p) = X^2 - (h + h^p)X + h^{p+1} = X^2 - \text{Tr}_p(h)X + 1$$

where  $\text{Tr}_p(h) \in \mathbb{F}_p$  denotes the trace of  $h$  over  $\mathbb{F}_p$ . Hence  $h$  can be represented by the polynomial  $P_h \in \mathbb{F}_p[X]$  and this polynomial is completely determined by the value of  $\text{Tr}_p(h)$ . Thus only  $\log p$  bits are required to represent elements of  $G_{q,p,2}$  by their minimal polynomials, compared to the  $2 \log p$  bits that would be required using a standard representation.

As already observed  $P_h$  does not determine  $h$  uniquely but determines both  $h$  and  $h^p$ , the conjugate of  $h$  over  $\mathbb{F}_p$ .

LUCDIF is a variant of Diffie-Hellman key exchange obtained by applying LUC to the conventional system described in the Introduction. In the LUCDIF variant Alice sends Bob  $\text{Tr}_p(g^a)$  instead of  $g^a$ . Using the standard method for solving a quadratic equation Bob solves  $X^2 - \text{Tr}_p(g^a)X + 1 = 0$  obtaining the solutions  $g^a$  and its conjugate  $g^{ap}$ . Bob can now use these solutions and his secret exponent  $b$  to calculate  $(g^a)^b + (g^{ap})^b = g^{ab} + g^{abp} = \text{Tr}_p(g^{ab})$ . Alice uses the same method to calculate the shared secret key  $\text{Tr}_p(g^{ab})$  from the value  $\text{Tr}_p(g^b)$  received from Bob.

The elements  $\text{Tr}_p(g^a)$  and  $\text{Tr}_p(g^b)$  that are communicated over the open channel are in  $\mathbb{F}_p$  and hence of length  $\log p$  bits. This is half the size of the elements  $g^a$  and  $g^b$  that are exchanged in conventional Diffie-Hellman key exchange using the standard representation discussed at the beginning of this section.

Another benefit of LUCDIF is that the calculations that each party must perform are significantly quicker than in the conventional system. These calculations use so-called Lucas recurrent sequences. For full details the reader should consult [2], [22] (where the name 'LUCDIF' was proposed), [17], [16], [19] and [14].

Of course it is essential that the benefits achieved by applying LUC do not compromise the security of the system. In fact it is easily shown that breaking the LUCDIF variant is equivalent to breaking the conventional system.

*Example 2.* The XTR system represents elements of a cyclotomic subgroup  $G_{q,p,6}$  by their minimal polynomials over  $\mathbb{F}_{p^2}$ . (This subgroup is called an XTR group in the XTR literature.) The minimal polynomial over  $\mathbb{F}_{p^2}$  of a non-identity element  $h \in G_{q,p,6}$  is

$$\begin{aligned} P_h &= (X - h)(X - h^{p^2})(X - h^{p^4}) \\ &= X^3 - (h + h^{p^2} + h^{p^4})X^2 + (h^{p^2+1} + h^{p^4+1} + h^{p^4+p^2})X - h^{p^4+p^2+1} \\ &= X^3 - \text{Tr}_{p^2}(h)X^2 + (h^{p^2+1} + h^{p^4+1} + h^{p^4+p^2})X - h^{p^4+p^2+1} \end{aligned}$$

where  $\text{Tr}_{p^2}$  denotes the trace over  $\mathbb{F}_{p^2}$ .

Since  $q \mid \Phi_6(p) = p^2 - p + 1$  and  $p^2 - p + 1 \mid p^4 + p^2 + 1$  we know that the constant term of  $P_h$  is  $-1$ . Furthermore the congruences  $p^2 + 1 \equiv p$ ,  $p^4 + 1 \equiv p^5$  and  $p^4 + p^2 \equiv p^3$  modulo  $p^2 - p + 1$  imply that the coefficient  $h^{p^2+1} + h^{p^4+1} + h^{p^4+p^2}$  is equal to  $\text{Tr}_{p^2}(h)^p$ . Thus

$$P_h = X^3 - \text{Tr}_{p^2}(h)X^2 + \text{Tr}_{p^2}(h)^p X - 1,$$

which is completely determined by the value of  $\text{Tr}_{p^2}(h) \in \mathbb{F}_{p^2}$ . It follows that only  $2 \log p$  bits are required to represent elements of  $G_{q,p,6}$  by their minimal polynomials, which compares very favourably to the  $6 \log p$  bits that would be required using a standard representation.

Clearly in order to apply XTR to a DL-based cryptosystem it is necessary to be able to perform certain computations using traces of elements of the XTR group. For example in Diffie-Hellman key exchange we must be able to compute  $\text{Tr}_{p^2}(g^{xy})$  given  $\text{Tr}_{p^2}(g^x)$  and  $y$ . Efficient methods for performing the calculations required for XTR variants of cryptosystems such as Diffie-Hellman key exchange and DSA have been developed by Lenstra and Verheul (see [10], [11], [12], [13]) and Lenstra and Stam (see [23]). As with LUC these methods are computationally more efficient than the corresponding calculations performed in  $G_{q,p,6}$  without using traces.

We conclude this section by discussing some security issues.

The most effective known methods of (passive) attack against DL-systems are based on the Birthday Paradox or use of the Number Field Sieve. Birthday Paradox based algorithms (such as Pollard's rho algorithm [20]) have expected running times of order  $\sqrt{q}$  elementary operations in  $G$ , where  $q$  is the largest prime factor of the order of  $G$ . The Discrete Logarithm variant of the Number Field Sieve has a heuristic expected asymptotic running time of  $L[p, 1/3, 1.923 + o(1)]$  (see [1] and [9]).

The security of the original Diffie-Hellman system, which uses  $G_{p,1}$ , depends not only on the size of  $p$  but also on that of the largest prime factor of  $p - 1$ ;

for adequate security this factor should have at least 160 bits. To resist Number Field Sieve attacks  $p$  should be at least 1024-bit.

For systems (like LUC, XTR) employing cyclotomic subgroups  $G_{q,p,k}$  of  $\mathbb{F}_{p^k}$  the same requirements on the size of  $q$  apply:  $q$  should have at least 160 bits to be secure against Birthday Paradox attacks. The following lemma (see also [9, Lemma 2.4], as corrected by Minghua Qu) shows that the condition  $q \nmid k$  ensures that every non-identity element of a cyclotomic subgroup  $G_{q,p,k}$  lies outside every proper subfield of  $\mathbb{F}_{p^k}$ , and hence that  $G_{q,p,k}$  is as secure against Number Field Sieve attacks as  $\mathbb{F}_{p^k}$  itself. This means that  $p$  should be chosen in such a way that  $k \cdot \log p > 1024$ . Hence for LUC  $p$  of at least 512 bits is recommended, and for XTR of at least 171 bits.

**Lemma 1.** *If  $h \in G_{q,p,k} \setminus \{1\}$  then  $h \notin \mathbb{F}_{p^d}$  for proper divisors  $d$  of  $k$ .*

*Proof.* Since  $q \nmid k$  we have  $\gcd(X^k - 1, kX^{k-1}) = 1$  in  $\mathbb{F}_q[X]$  and thus  $X^k - 1$  has no repeated roots in the algebraic closure of  $\mathbb{F}_q$ . As  $X^k - 1 = \prod_{e|k} \Phi_e(X)$  and  $\Phi_k(p) \equiv 0 \pmod q$  we see that  $\Phi_e(p) \not\equiv 0 \pmod q$  for  $e \mid k$ ,  $e < k$ . But for any proper divisor  $d$  of  $k$  we have

$$X^d - 1 = \prod_{e|d} \Phi_e(X) \mid \prod_{\substack{e|k \\ e < k}} \Phi_e(X),$$

so  $p^d - 1 \not\equiv 0 \pmod q$ . Thus the order of  $\mathbb{F}_{p^d}^*$  is not a multiple of the order  $q$  of  $h$ .

### 3 Do More Compact Representations than XTR Exist?

By representing elements of cyclotomic subgroups by their minimal polynomials over a subfield, LUC and XTR reduce the number of required bits per element by a factor 2 and 3 respectively. A natural question arises: can we do any better?

Both LUC and XTR provide evidence for the BPV conjecture mentioned in the Introduction, which can be informally stated as follows, using Euler's totient function  $\phi$ :

*Elements of  $G_{q,p,k}$  can be represented with  $\phi(k) \log p$  bits using minimal polynomials over some subfield of  $\mathbb{F}_{p^k}$ .*

If the BPV conjecture were true the best size reduction (compared to a standard representation) is achieved when the ratio  $k/\phi(k)$  is large. This happens when  $k$  is the product of distinct primes. LUC and XTR are the simplest such cases and the next value of  $k$  of interest would be  $k = 2 \cdot 3 \cdot 5 = 30$ . We shall investigate this case in Section 7.

We now present two more examples that provide further evidence in support of the BPV conjecture.

*Example 3.* Let  $k$  be a prime and let  $h \in G_{q,p,k}$ , with  $h \neq 1$ . The minimal polynomial  $P_h$  of  $h$  over  $\mathbb{F}_p$  has degree  $k$  and constant term equal to 1 if  $k = 2$  and  $-1$  otherwise (see Theorem 4). Therefore  $P_h$  is completely determined by

the  $k - 1$  coefficients of  $X, X^2, \dots, X^{k-1}$ . Since these coefficients are elements of  $\mathbb{F}_p$  and  $\phi(k) = k - 1$  it follows that elements of  $G_{q,p,k}$  can be represented by  $\phi(k) \log p$  bits, in support of the BPV conjecture.

Note that one can base generalisations of LUC on this example. In fact such a variant was published by G. Gong and L. Harn for  $k = 3$  (see [7]). Here recurrent Lucas sequences similar to those used in LUC are employed. However this system has an ‘improvement factor’  $k/\phi(k)$  of just  $3/2$ .

*Example 4.* Let  $k = 6$ , so that the extension field has the same degree as in XTR (Example 2). In XTR we considered the minimal polynomial over  $\mathbb{F}_{p^2}$  of  $h \in G_{q,p,6}$ ,  $h \neq 1$ . We now consider the minimal polynomial

$$P_h = \prod_{0 \leq i \leq 5} (X - h^{p^i}) = X^6 + a_5 X^5 + a_4 X^4 + a_3 X^3 + a_2 X^2 + a_1 X + a_0$$

of  $h$  over  $\mathbb{F}_p$ . The constant term  $a_0 = 1$  since the order  $q$  of  $h$  divides  $\Phi_6(p)$  which in turn divides  $1 + p + p^2 + p^3 + p^4 + p^5$ . Using  $p^3 \equiv -1 \pmod{q}$  it is easily shown that  $a_1 = a_5$  and  $a_2 = a_4$  (cf. Corollary 11). We note that the value of the first elementary symmetric polynomial in the conjugates of  $h$  is  $-a_5$  and the value of the second elementary symmetric polynomial in the conjugates of  $h$  is  $a_4$ . Furthermore one can write  $a_3$  (which is minus the value of the third elementary symmetric polynomial in the conjugates of  $h$ ) as a symmetric polynomial of degree 2 in the conjugates of  $h$ . By the Fundamental Theorem of Symmetric Polynomials [18, Theorem 4.31] it follows that it is possible to write  $a_3$  as a polynomial in  $a_5$  and  $a_4$ . In fact we have  $a_3 = -a_5^2 + 2a_4 + 2a_5 - 2$ , a relationship first noted in [4]. It follows that  $P_h$  is completely determined by  $a_5$  and  $a_4$  so that  $h$  can be represented by two elements of  $\mathbb{F}_p$ , that is by  $\phi(6) \log p$  bits, in support of the conjecture.

The four examples that we have considered so far have demonstrated relationships that can hold between coefficients of minimal polynomials of elements of a cyclotomic subgroup  $G_{q,p,k}$ . In the next section we shall prove some general results concerning these relationships. We shall also formulate a weaker version of the BPV conjecture (Conjecture 3) that is more amenable to verification.

## 4 Coefficients of the Minimal Polynomials

We begin with the following theorem.

**Theorem 1.** *Let  $h$  be a generator of a cyclotomic subgroup  $G_{q,p,k}$ , where  $p$  is odd and  $k \geq 2$ . Let  $X^{k/d} + a_{k/d-1}X^{k/d-1} + \dots + a_1X + a_0$  be the minimal polynomial of  $h$  over  $\mathbb{F}_{p^d}$ , for some  $d$  dividing  $k$ , with  $d < k$ . Then  $a_0 = (-1)^{k/d}$ , and if  $k = 2\ell$  is even,  $a_i = (-1)^{k/d} a_{k/d-i}^{p^\ell}$ , for  $i = 1, \dots, k/d - 1$ .*

*Proof.* Write  $h_j = h^{p^{dj}}$  for  $j = 0, \dots, k/d - 1$ . Then

$$X^{k/d} + a_{k/d-1}X^{k/d-1} + \dots + a_1X + a_0 = \prod_{j=0}^{k/d-1} (X - h_j),$$

and comparing coefficients we see that  $a_i = (-1)^{k/d-i} \sigma_{k/d-i}(h_0, \dots, h_{k/d-1})$  for  $i = 0, \dots, k/d-1$ , where  $\sigma_n(h_0, \dots, h_{k/d-1})$  is the  $n$ -th elementary symmetric polynomial in the conjugates  $h_j$  of  $h$ . In particular

$$\begin{aligned} a_0 &= (-1)^{k/d} \sigma_{k/d}(h_0, \dots, h_{k/d-1}) \\ &= (-1)^{k/d} h_0 \cdots h_{k/d-1} = (-1)^{k/d} h^{1+p^d+p^{2d}+\dots+p^{k-d}}. \end{aligned}$$

But  $1+p^d+p^{2d}+\dots+p^{k-d} = (p^k-1)/(p^d-1)$  which is divisible by  $\Phi_k(p)$  and hence by  $q$ , the order of  $h$ . Therefore  $a_0 = (-1)^{k/d}$ .

If  $k = 2\ell$  is even then  $p^k - 1 = (p^\ell - 1)(p^\ell + 1)$ . Since the order  $q$  of  $h$  divides  $p^k - 1$  but not  $p^\ell - 1$  we have  $p^\ell \equiv -1 \pmod{q}$  and therefore  $h_j^{-1} = h_j^{p^\ell}$  for  $j = 0, \dots, k/d-1$ . Furthermore, since  $h_0 \cdot h_1 \cdots h_{k/d-1} = 1$  we have  $\sigma_{k/d-i}(h_0, \dots, h_{k/d-1}) = \sigma_i(h_0^{-1}, \dots, h_{k/d-1}^{-1})$  for  $i = 1, \dots, k/d-1$ . It follows that for  $i = 1, \dots, k/d-1$

$$\begin{aligned} \sigma_{k/d-i}(h_0, \dots, h_{k/d-1}) &= \sigma_i(h_0^{-1}, \dots, h_{k/d-1}^{-1}) = \sigma_i(h_0^{p^\ell}, \dots, h_{k/d-1}^{p^\ell}) \\ &= \sigma_i(h_0, \dots, h_{k/d-1})^{p^\ell} \quad (\text{characteristic } p) \\ &= ((-1)^i a_{k/d-i})^{p^\ell} = (-1)^i a_{k/d-i}^{p^\ell}. \end{aligned}$$

Therefore, as required, for  $i = 1, \dots, k/d-1$ :

$$a_i = (-1)^{k/d-i} \sigma_{k/d-i}(h_0, \dots, h_{k/d-1}) = (-1)^{k/d} a_{k/d-i}^{p^\ell}.$$

**Corollary 1.** *If  $k$  is even and  $d$  divides  $k/2$  then the minimal polynomial over  $\mathbb{F}_{p^d}$  of a generator of  $G_{q,p,k}$  is palindromic:  $a_i = a_{k/d-i}$  for  $i = 0, \dots, k/d$ .*

*Proof.* Write  $\ell = k/2$ . Elements of  $\mathbb{F}_{p^d}$  are invariant under  $p^\ell$ -th powering since  $d$  divides  $\ell$ . Hence, by the previous theorem,  $a_i = (-1)^{k/d} a_{k/d-i}$  for  $i = 0, \dots, k/d$ . Since  $k/d$  is even the result follows.

**Proposition 1.** *Let  $k = de$ , with  $e > 1$ . Then for any element  $h$  of  $G_{q,p,k}$  the minimal polynomial  $P_h$  over  $\mathbb{F}_{p^d}$  can be represented using the following number of elements of  $\mathbb{F}_{p^d}$ :*

- $e - 1$ , if  $de$  is odd;
- $\frac{e-1}{2}$ , if  $d$  is even and  $e$  is odd;
- $\frac{e}{2}$  if  $e$  is even.

*Proof.* We represent elements of  $G_{q,p,k}$  by their minimal polynomials over the subfield of degree  $d$ . The constant coefficient is  $\pm 1$ , so  $e-1$  elements of  $\mathbb{F}_{p^d}$  suffice to represent elements of  $G_{q,p,k}$ . This covers the first case.

In the second and third cases  $k$  is even and by Theorem [1](#) only half of the remaining  $e-1$  coefficients are required. More precisely if  $e$  is odd we need  $(e-1)/2$  coefficients and if  $e$  is even we require  $e/2$  coefficients.



Note that, unfortunately, this result cannot be used recursively since the coefficients  $a_i$  are not (in general) in a cyclotomic subgroup of  $\mathbb{F}_{p^d}$ .

Proposition 1 leaves a choice for  $d$  and  $e$  if  $k$  is composite, and some choices will offer better improvement factors than others. If  $k$  is even but not a power of two then Proposition 1 indicates that a good choice for  $e$  is the smallest odd divisor of  $k$  greater than 1. For example when  $k$  is divisible by 6 we can choose  $e = 3$  and  $d = k/3$  and thereby achieve an improvement ratio of 3. In the following example, which generalises Example 4, we show that the same improvement ratio can be achieved by taking  $e = 6$ .

*Example 5.* Let  $k$  be of the form  $6d$ , let  $r = p^d$  and consider  $P_h$ , the minimal polynomial over  $\mathbb{F}_r$  of  $h \in G_{q,p,k} \setminus \{1\}$ :

$$P_h = \prod_{0 \leq i \leq 5} (X - h^{r^i}) = X^6 + a_5 X^5 + a_4 X^4 + a_3 X^3 + a_2 X^2 + a_1 X + a_0$$

where  $a_i \in \mathbb{F}_r$ .

The order of  $h$  is  $q$ , which divides  $\Phi_{6d}(p)$ . It is well-known (see [8]) that  $\Phi_{6d}(X) \mid \Phi_6(X^d)$  so  $q$  divides  $\Phi_6(r) = r^2 - r + 1$ .

Arguing as in Example 2 we have

$$(X - h)(X - h^{r^2})(X - h^{r^4}) = X^3 - tX^2 + t^r X - 1,$$

where  $t = \text{Tr}_{r,2}(h) \in \mathbb{F}_{r^2}$  is the trace of  $h$  over  $\mathbb{F}_{r^2}$ . From this we have

$$(X - h^r)(X - h^{r^3})(X - h^{r^5}) = X^3 - t^r X^2 + t^{r^2} X - 1.$$

Since  $t^{r^2} = t$  it follows that

$$P_h = (X^3 - tX^2 + t^r X - 1)(X^3 - t^r X^2 + tX - 1),$$

and we see that not only is  $P_h$  palindromic, as Corollary 1 implies, with  $a_0 = 1$ ,  $a_1 = a_5 = -t - t^r$  and  $a_2 = a_4 = t + t^r + t^{1+r}$ , but also that

$$\begin{aligned} a_3 &= -2 - t^2 - t^{2r} \\ &= -(-t - t^r)^2 + 2(t + t^r + t^{1+r}) + 2(-t - t^r) - 2 \\ &= -a_5^2 + 2a_4 + 2a_5 - 2. \end{aligned}$$

This means that we only need  $a_5$  and  $a_4$  to specify  $P_h$ .

**Table 1.** The table summarises the results of Proposition 1 and Example 5 concerning the number of words  $S$  of size  $\log p$  that suffices to represent the minimal polynomials of elements of  $G_{q,p,de}$ , and the improvement ratio  $de/S$ . Note that  $S$  is an upper bound; fewer words may do.

$d$	$e$	$S$	ratio: $de/S$
odd	odd	$d \cdot (e - 1)$	$e/(e - 1)$
even	odd	$d \cdot \frac{e-1}{2}$	$2e/(e - 1)$
even	even	$d \cdot \frac{e}{2}$	2
odd	even	$d \cdot \frac{e}{2}$	2
any	6	$d \cdot 2$	3

## 5 The Conjectures

We now work towards a more precise formulation of the BPV conjecture. Consider some  $k = de$  with  $e > 1$ . Let  $h$  be an element of the  $(p, k)$ -cyclotomic group  $G_{p,k}$  that is not contained in any proper subfield of  $\mathbb{F}_{p^k}$  and let  $P_h^{(d)} = X^e + a_{e-1}X^{e-1} + \dots + a_1X + a_0$  be the minimal polynomial of  $h$  over  $\mathbb{F}_{p^d}$ . Note that  $a_{e-j}$  corresponds, up to sign, to the  $j$ -th elementary symmetric polynomial evaluated in the  $e$  conjugates of  $h$  over  $\mathbb{F}_{p^d}$ . By Theorem [1](#) we have  $a_0 = (-1)^e$ .

For  $1 \leq i \leq e-1$  let  $A_i = \{a_{e-1}, \dots, a_{e-i}\}$ . We let  $u_d$  denote the smallest integer with the property that the set  $A_{e-1}$  of all non-trivial coefficients of  $P_h^{(d)}$  can be recovered from  $A_{u_d}$ . Thus all coefficients of  $P_h^{(d)}$  can be recovered from the first  $u_d$  elementary symmetric polynomials in the conjugates of  $h$  over  $\mathbb{F}_{p^d}$  but *not* from the first  $u_d - 1$  polynomials.

We must address the question of what we mean by ‘recovering’  $A_{e-1}$  from a subset  $A_i$ . Note that we should not simply state that all  $a_j$  can be expressed as polynomials in the elements of  $A_i$ , since the coefficients come from a finite field in which many relations will exist. It seems one requires the existence of such an expression *independent of  $p$* , although perhaps dependent on  $d$  and  $e$ . However this is still not entirely satisfactory: the second part of Theorem [1](#) states that we can recover, for example,  $a_1$  from  $a_{k/d-1}$  using conjugates, i.e. in a manner which *does* depend on  $p$ . This means that our ‘recovery’ notion for  $d > 1$  should imply the existence of polynomials with integer coefficients and degree independent of  $p$  that, when evaluated in the  $d$  conjugates over  $\mathbb{F}_p$  of the elements of  $A_i$ , will yield the other coefficients.

We shall introduce multivariate polynomials in indeterminates  $X_j$ , and evaluate the polynomials at the elements of some coefficient set  $A_i$ . It will be convenient to define the *weighted degree* of a monomial  $X_1^{e_1} \dots X_n^{e_n}$  in  $\mathbb{Z}[X_1, \dots, X_n]$  to be  $\sum_{j=1}^n j \cdot e_j$  and the weighted degree of a polynomial  $P$  as the maximum of the weighted degrees of the monomials that appear in  $P$  (with non-zero coefficient). Note that  $X_j$  has weighted degree  $j$  in  $P$ . The motivation for this definition is that we shall evaluate  $X_j$  in  $a_{e-j}$ , which is symmetric of degree  $j$  in the conjugates of  $h$  over  $\mathbb{F}_{p^d}$ .

Observe that  $G_{p,k}$  is asymptotically of size  $p^{\phi(k)}$ . Therefore in order to represent the whole of  $G_{p,k}$  by the minimal polynomials of its elements over  $\mathbb{F}_{p^d}$  we must have  $d \cdot u_d \geq \phi(k)$ . Thus, for given values of  $k$  and  $d$ , we have an information-theoretic lower bound of  $\lceil \phi(k)/d \rceil$  on the value of  $u_d$ . The conjecture states that in fact  $u_d$  is always *equal* to this lower bound.

We now come to our first formulation of the BPV conjecture.

**Conjecture 1 (( $d, e$ )-BPV).** *Let  $k = de$ , with  $e > 1$ . Let  $u_d$  be the least value of  $u$  for which  $Q_j \in \mathbb{Z}[X_1^{(0)}, \dots, X_1^{(d-1)}, X_2^{(0)}, \dots, X_2^{(d-1)}, \dots, X_u^{(0)}, \dots, X_u^{(d-1)}]$  exist, for  $1 \leq j \leq e - u - 1$ , such that for every prime  $p$  and every element  $h \in G_{p,k}$  that is not contained in a proper subfield of  $\mathbb{F}_{p^k}$ , the coefficient  $a_j$  of  $P_h^{(d)}$  is given by*

$$a_j = \bar{Q}_j(a_{e-1}, a_{e-1}^p, \dots, a_{e-1}^{p^{d-1}}, a_{e-2}, a_{e-2}^p, \dots, a_{e-2}^{p^{d-1}}, \dots, a_{e-u}, a_{e-u}^p, \dots, a_{e-u}^{p^{d-1}}),$$

for  $1 \leq j \leq e - u - 1$ , where  $\bar{Q}_j$  denotes  $Q_j$  with coefficients taken modulo  $p$ . Then  $u_d = \lceil \phi(de)/d \rceil$ .

Motivated by Example 4 we also formulate a strong form of the conjecture, including a bound on the (weighted) degree of the polynomials involved.

**Conjecture 2 (strong  $(d, e)$ -BPV).** *Let  $k = de$ , with  $e > 1$ . Let  $u_d^*$  be the smallest integer for which there exist polynomials  $Q_j$  as in Conjecture  $(d, e)$ -BPV with the additional requirement that the polynomials  $Q_j$  are of weighted degree at most  $u$ , where the weighted degree of  $X_k^{(i)}$  is  $k$  (for  $1 \leq k \leq u$  and  $0 \leq i \leq d - 1$ ). Then  $u_d^* = \lceil \phi(de)/d \rceil$ .*

The main conjecture, which was stated informally in Section 3, can now be made precise. For  $k > 1$  we define  $\text{red}(k) = \min\{d \cdot u_d\}$ , where the minimum is taken over all proper divisors  $d$  of  $k$ .

**Conjecture 3 ( $k$ -BPV).** *Let  $k > 1$  be an integer. There exists a proper divisor  $d$  of  $k$  such that  $d$  divides  $\phi(k)$  and for which  $(d, k/d)$ -BPV holds. Therefore  $\text{red}(k) = \phi(k)$ .*

Conjecture 3 applies to all elements of  $G_{p,k}$  that are not contained in a proper subfield of  $\mathbb{F}_{p^k}$ . Therefore if Conjecture 3 were true then the BPV conjecture, which we expressed earlier in terms of cyclotomic subgroups  $G_{q,p,k}$ , would certainly hold as well.

Finally, we formulate the obvious strengthened version of Conjecture 3, including a bound on the degree.

**Conjecture 4 (strong  $k$ -BPV).** *Let  $k > 1$  be an integer. There exists a proper divisor  $d$  of  $k$  such that  $d$  divides  $\phi(k)$  and for which strong  $(d, k/d)$ -BPV holds.*

Our preparatory work on the coefficients implies the correctness of Conjecture 4 and hence of Conjecture 3 for a whole family of values of  $k$ .

**Proposition 2.** *Let  $2^s p_1^{r_1} p_2^{r_2} \dots p_n^{r_n}$  be the prime factorisation of  $k > 1$  with  $2 < p_1 < \dots < p_n$ . Then  $\text{red}(k) \leq \phi(2^s p_1^{r_1}) p_2^{r_2} \dots p_n^{r_n}$ . In particular, if  $k$  is of the form  $2^s p_1^{r_1}$  then Conjecture 4 holds for  $k$ ; and  $\text{red}(k) = \phi(k)$  in this case.*

*Proof.* If  $k = 2^s$  and  $s \geq 1$  then taking  $d = 2^{s-1}$  and  $e = 2$  in the even-even case of Proposition 1 gives the result since  $\phi(2^s) = 2^{s-1}$ . Similarly, if  $k = p_1^{r_1}$  with  $r_1 \geq 1$ , the result follows from taking  $d = p_1^{r_1-1}$  and  $e = p_1$  in the odd-odd case as  $\phi(p_1^{r_1}) = (p_1 - 1)p_1^{r_1-1}$ . The general case of the first part of the result (where  $r_1, s \geq 1$ ) follows by taking  $d = 2^s p_1^{r_1-1} p_2^{r_2} \dots p_n^{r_n}$  and  $e = p_1$  in the even-odd case. The final part follows directly, using the observation that  $\text{red}(k) \geq \phi(k)$ .

Proposition 2 implies that Conjecture 4 holds for all  $k \leq 30$  with  $k \neq 15, 21, 30$ .

*Example 6.* The first case of real interest of Conjecture  $k$ -BPV is  $k = 30$ , as there an improvement ratio of  $30/8 > 3$  might be obtained. By virtue of Proposition 1 it follows that, by choosing  $e = 3$ , we can represent elements of the cyclotomic group  $G_{p,30}$  as minimal polynomials using a single coefficient from the subfield of  $p^{10}$  elements, so using  $10 \log p$  bits. Therefore  $\text{red}(30) \leq 10$ .

Conjecture 30-BPV states that in fact only  $\phi(30) \log p = 8 \log p$  bits are necessary. More specifically the conjecture says that for some divisor  $d$  of  $\phi(30) = 8$ , the minimal polynomial over  $\mathbb{F}_{p^d}$  of any element of  $G_{p,30}$  can be generated by eight elements of  $\mathbb{F}_p$ . For  $d = 2$ , for example, it could be that only the four highest of the non-trivial coefficients of minimal polynomials over  $\mathbb{F}_{p^2}$  are independent, and that the others can be expressed as polynomial expressions in these. This would represent a significant improvement on the upper bound provided by Proposition 1, which states that the seven highest non-trivial coefficients are sufficient to generate the others.

The 25 pairs  $(d, e)$  with  $de \leq 30$  for which Proposition 1 and Example 5 do not provide a proof of Conjecture 2 are listed in the table at the end of Section 7.

## 6 The Magma Programs

In order to test the conjectures formulated in the previous section we performed some experiments using the computer algebra system MAGMA [3].

**Algorithm 0** (Find relations). *Input:* integers  $p, k, d, u, v, j$ .

*Output:* a set  $\mathcal{Q}$  of polynomials in  $\mathbb{Z}[X_1, \dots, X_u, Y]$ .

*Description:*

Determine a prime divisor  $q$  of  $\Phi_k(p)$  not dividing  $k$  (Lemma 1), and a generator  $h$  of  $G_{q,p,k}$  (e.g. taking the  $(p^k - 1)/q$ -th power of a primitive element  $g$  of  $\mathbb{F}_{p^k}$ ).

Next, generate the finite set  $S$  of all sequences  $[s_1, \dots, s_u]$  with  $\sum_{i=1}^u i \cdot s_i \leq v$ .

Now generate  $s = \#S$  random elements  $h_1, \dots, h_s$  of  $G_{q,p,k}$ , for example by taking random powers of  $h$ , determine the minimal polynomials

$$P_{h_i}^{(d)} = X^{k/d} + a_{k/d-1} X^{k/d-1} + \dots + a_1 X + a_0,$$

of these elements over  $\mathbb{F}_{p^d}$  and evaluate  $m(h_i, \mathbf{s}) = a_{e-1}^{s_1} \cdot a_{e-2}^{s_2} \cdot \dots \cdot a_{e-u}^{s_u}$  for all  $\mathbf{s} \in S$ . Let  $M$  be the square  $s \times s$  matrix with entries in  $\mathbb{F}_{p^d}$ , the  $i$ -th row of which consists of the monomials  $m(h_i, \mathbf{s})$ , for  $\mathbf{s}$  ranging over  $S$ . Let  $\mathbf{w} \in \mathbb{F}_{p^d}^s$  consist of the coefficients  $a_j$  (with  $j$  given by the input) of the polynomials  $P_{h_i}^{(d)}$ , for  $i = 1, 2, \dots, s$ .

Solve the linear system of equations  $M\mathbf{c} = \mathbf{w}$  for  $\mathbf{c} \in \mathbb{F}_p^s$ . If the solution space is non-empty, translate each element  $\mathbf{c}$  from the solution space back to a polynomial relation  $C \in \mathbb{Z}[X_1, \dots, X_u]$  via

$$\mathbf{c} \mapsto C = \sum_{\mathbf{s} \in S} c_{\mathbf{s}} X_1^{s_1} \dots X_u^{s_u} - Y,$$

where on the right we interpret the component  $c_{\mathbf{s}} \in \mathbb{F}_p$  of the vector  $\mathbf{c}$  as an integer by taking the least integer representative for its residue class modulo  $p$ .

Finally, determine the Gröbner basis  $\mathcal{Q}$  of the ideal generated by these relations in  $\mathbb{Q}[X_1, \dots, X_u, Y]$ .

This ends the description of the algorithm.

The output of the algorithm consists of polynomials in  $u$  variables that form a basis for all polynomial relations  $Q(a_{e-1}, \dots, a_{e-u}) - a_j = 0$  between the coefficients of the minimal polynomial for generators of the cyclotomic subgroup  $G_{q,p,k}$ , satisfying the condition that the weighted degree of  $Q$  is at most  $v$ .

To verify Conjecture 2 for a pair  $(d, e)$  we apply the following algorithm, with input  $d, e$  and with  $w = \lceil \phi(de)/d \rceil$ .

**Algorithm 2.** *Input: integers  $d, e, w$ .*

*Output: either ‘false’ or sets  $\mathcal{Q}_j$  of candidate polynomials for Conjecture 2.*

*Description:*

Let  $u = \lceil \phi(de)/d \rceil$ .

Repeat the following step for  $j = e - u - 1, e - u - 2, \dots, 1$  in succession, terminating with output ‘false’ when an empty set  $\mathcal{Q}_j$  is encountered, and with sets  $\mathcal{Q}_j$ ,  $j = e - u - 1, \dots, 1$ , as output otherwise:

Choose a prime number  $p$ , and apply Algorithm 1 with input  $p, k = de, d, u, v = w, j$  to determine a set  $\mathcal{Q}_j$ .

If Algorithm 2 returns ‘false’, Conjecture 2 is refuted for the given values of  $d, e$  as no polynomial relation exists (for at least one  $j$ ) of weighted degree at most  $u_d$  that works modulo  $p$ . Otherwise it returns candidate polynomials  $Q_j$  expressing  $a_j$  in  $a_{e-1}, \dots, a_{e-u}$ . These candidates have only been proven to work for a single prime number  $p$ ; to increase confidence one would test the candidates for different values of  $p$ .

A (less effective) alternative to Algorithm 2 consists of a *single* application of Algorithm 1 rather than  $e - u - 1$  successive ones, by replacing in the input for Algorithm 1 the values of  $u$  and  $v$  by  $e - 1$ , and putting  $j = 0$ . The result will be that Algorithm 1 will attempt to find all algebraic relations between all  $a_i$ ’s (of weighted degree bounded by  $v$ ) in one go; if the Conjectured relations exist, the Gröbner basis will exhibit them all. This approach is only feasible for very small values of  $de$  (see Example 7).

Algorithm 2 rarely succeeds; it is designed to refute Conjecture 2 for pairs  $d, e$ . Likewise, the following algorithm is designed to refute Conjecture 1.

**Algorithm 1.** *Input: integers  $d, e$ .*

*Output: either ‘false’ or sets  $\mathcal{Q}_j$  of candidate polynomials for Conjecture 1.*

*Description:*

Repeatedly apply Algorithm 2 with input triples  $d, e, w$ , until sets  $\mathcal{Q}_j$  are returned, starting with  $w = \lceil \phi(de)/d \rceil$ , and incrementing  $w$  by 1 when Algorithm 2 returns ‘false’.

It should now also be clear how to attempt to refute (or prove) Conjectures 3, 4: apply Algorithms 1, 2 for all pairs  $(d, e)$  of divisors of  $k$  with  $d$  dividing  $\phi(k)$ .

As stated, the algorithms do not look for dependencies involving the  $\mathbb{F}_p$ -conjugates of  $a_{e-1}, \dots, a_{e-u}$ . The reason for this is that initially we attempt to find dependencies that do not involve the proper conjugates; the algorithm can easily be modified to include them, but doing this blows up the number of variables in the monomials by a factor  $d^u$ . We have omitted this from the description of the algorithms for the sake of clarity.

The Gröbner basis of the ideal is determined to detect dependencies between relations that are found. In our experiments usually one of three things happened: either no relation was found, or many relations were found due to the fact that the prime was chosen too small, or a few relations were found that generated an ideal with a Gröbner basis consisting of a single polynomial relation expressing the dependency of  $a_{e-u-1}$  on  $a_{e-1}, \dots, a_{e-u}$ . See [5], for example, for a discussion of Gröbner bases.

The main feature of Algorithm 0 is that of converting the problem of finding a polynomial relation to finding the kernel of a matrix over a finite field: the columns of the matrix correspond to the monomials, and the existence of an algebraic relation between the coefficients implies a dependency between the evaluations of the monomials at the coefficients, that is between the columns of the matrix. Thus the problem is reduced to linear algebra over  $\mathbb{F}_{p^d}$ .

The experiments we carried out deviated slightly from the description in this section: the indeterminate  $Y$  in Algorithm 0 was given weight  $u + 1$  and was allowed to appear with exponent larger than 1 in the polynomial relations (see Example 10). For that reason our searches started at weight (a multiple of)  $\lceil \phi(de)/d \rceil + 1$ , exceeding the minimal value predicted by Conjecture 2.

*Example 7.* As a first example we present the output of our algorithm for  $k = 4$ .

Conjecture 1 holds for  $(d, e) = (2, 2)$  since by Theorem 1 the minimal polynomials over  $\mathbb{F}_{p^2}$  of elements of  $G_{p,4} \setminus \mathbb{F}_{p^2}$  are of the form  $X^2 + a_1X + 1$ , with  $a_1 \in \mathbb{F}_{p^2}$ .

The other case for  $k = 4$  is  $d = 1, e = 4$ . The minimal polynomials over  $\mathbb{F}_p$  of elements of  $G_{p,4} \setminus \mathbb{F}_{p^2}$  are of the form  $X^4 + a_3X^3 + a_2X^2 + a_1X + a_0$ ,  $a_0 = 1$ .

In a run of Algorithm 2 we used  $p = 5$ ; since  $\Phi_4(5) = 5^2 + 1 = 26$ , we look at  $G_{13,5,4}$  in  $G_{5,4}$ . To find possible algebraic relations between  $a_3, a_2$  and  $a_1$  we take  $u = 2$ ; we choose  $w = 3$ . The only monomials besides  $Y$  we obtain are  $X_1^2, X_2, X_1, 1$ . One run of our algorithm produced two dependencies in the matrix  $M$ , corresponding to an ideal with Gröbner basis  $X_1 - Y, X_2 - Y^2 + Y + 1$ . The first of these expresses that  $a_3 = a_1$ , as we expect by Theorem 1 but the second is an ‘accident’ caused by the fact that we have chosen  $p$  and thereby  $q$  to be very small. Indeed, in this case several minimal polynomials coincided (since the 12 non-trivial elements have just 3 different minimal polynomials). This illustrates why small primes  $p$  should be avoided.

If we invoke the algorithm with  $p = 101$  instead, we immediately find a single relation  $a_3 - a_1 = 0$  and no others. As a matter of fact, if we increase the parameters  $u$  and  $w$  to 3 and 4, the result will be a Gröbner basis  $a_3 - a_1, a_0 - 1$ : the minimal polynomials are always palindromic, and we have rediscovered Corollary 1 for this case.

Refuting Conjecture 1 (without the degree bounds) would involve looking at evaluations of *all* possible monomials in  $u$  coefficients. But since there is only a finite number of (different powers of) elements in  $\mathbb{F}_{p^k}$  anyway, this is still a finite task! However, the necessary computation can only be done if  $p$  is very small (say 2 or 3), in which case we run into problems similar to those in the previous

example for small  $p$ , namely that the order of  $G_{p,k}$  would be smaller than the number of monomials and we would obtain many unwanted identities.

## 7 Experimental Results

In this section we describe the experiments we have performed to test the conjectures. We looked at all cases with  $k = de \leq 30$  still left open, as summarised in the table at the end of this section. We comment on some interesting cases, in order of ascending  $k$ .

*Example 8.*  $k = 6$

This provided a test case for our programs (see the earlier examples). There are three pairs  $(d, e)$  to consider.

The case  $d = 3, e = 2$  is trivial, since in the quadratic extension  $\mathbb{F}_{p^6}$  over  $\mathbb{F}_{p^3}$  elements are given by a single non-trivial element of the palindromic minimal polynomial; elements are thus represented by one element of  $\mathbb{F}_{p^3}$ , which is in accordance with Conjecture 1.

With  $d = 1, e = 6$  Algorithm 2 finds (within a few seconds) the relations  $-a_5^2 + 2a_4 + 2a_5 - 2 = a_3$ ,  $a_4 = a_2$  and  $a_5 = a_1$ , for example with  $p = 211$ . Thus elements of a degree six field can be represented by the elements  $a_5$  and  $a_4$  from the prime field. This proves, as noted before, both Conjecture 2 and hence Conjecture 1 for  $(d, e) = (1, 6)$ , as well as Conjecture 4 and hence Conjecture 3 for  $k = 6$ .

With  $d = 2, e = 3$  we have a cubic extension with  $k$  even. The standard Algorithm 2 does not find small relations; however, if we include the conjugates  $a_2^p$  and  $a_1^p$  as well as  $a_2$  and  $a_1$ , then with  $p = 29$  the algorithm produces the relation  $a_2 + a_1^p = 0$  (and also, in fact,  $a_1^{p^2} - a_1 = 0$  in the Gröbner basis). Note the conflicting constraints on  $p$  once we include the conjugates: we want  $p$  to be large to avoid spurious relation in a small field, whereas we want it to be small since we get monomials including  $a_i^{p^j}$  in the relations. The relation  $a_2 + a_1^p = 0$  means we can represent the degree 6 field by a single element  $a_1$  from the quadratic subfield; the element  $a_2$  can then be recovered. This proves Conjecture 1 for  $(d, e) = (2, 3)$  and Conjecture 3 for  $k = 6$  (again).

*Example 9.*  $k = 9$

Conjecture 2 and Conjecture 4 for  $d = e = 3$  are covered by Proposition 1.

To achieve the same efficiency in the full extension ( $d = 1, e = 9$ ) one would have to express  $a_2$  and  $a_1$  in terms of  $a_8, a_7, \dots, a_3$ . Our experiments with Algorithm 2 show that no such relations exist with  $u = 6$  and  $w = 7$ . That is, Conjecture 2 is false for this case.

In order to investigate whether a relaxation as in Conjecture 1 with regard to the degree of the polynomials involved would hold, we increased the search bound in Algorithm 1 to  $w = 28$ . In this and the cases to follow, we took  $w$  as a multiple of  $u+1$  and took  $w$  also as an upper bound on the weighted degree of the

polynomials when all variables are taken into consideration; that is, we searched for polynomials in  $\mathbb{Z}[X_1, \dots, X_u, Y]$  for which  $\sum_{i=1}^u i \cdot s_i + (u+1) \cdot s \leq w$ , where  $s$  is the exponent of  $Y$ . In the current case that simply means that we allowed polynomials involving  $Y$  up to the 4-th power. Thus we even consider relations for  $a_2$  involving  $a_2^4$ .

No relations were found with  $w = 28$ . The computation involved computing the kernel of a  $8561 \times 8561$  matrix over  $\mathbb{F}_p$  (using  $p = 2003$ ).

*Example 10.*  $\boxed{k = 10}$

The case  $d = 2, e = 5$  can be done using 2 elements from  $\mathbb{F}_{p^2}$  by Proposition [11](#) proving Conjecture [11](#) for  $(d, e) = (2, 5)$  but also Conjecture [13](#) for  $k = 10$ .

For  $d = 1, e = 10$  Proposition [11](#) shows that 5 elements of  $\mathbb{F}_p$  suffice; Conjecture [11](#) predicts that 4 should be enough. We therefore invoke the Algorithm with  $u = 4$ ; we found the following relation, but only after raising the search limit to  $w = 15$  (using  $p = 1009$ ):

$$\begin{aligned}
& a_9^8 + 2 \cdot a_9^7 \cdot a_8 - 8 \cdot a_9^6 \cdot a_8 - 2 \cdot a_9^6 \cdot a_7 + a_9^6 \cdot a_5 - 12 \cdot a_9^5 \cdot a_8 + 4 \cdot a_9^5 \cdot a_7 + 4 \cdot a_9^5 \cdot a_5 \\
& - 4 \cdot a_9^5 \cdot a_8 + 21 \cdot a_9^4 \cdot a_8^2 + 12 \cdot a_9^4 \cdot a_8 \cdot a_7 - 2 \cdot a_9^4 \cdot a_8 \cdot a_6 - 6 \cdot a_9^4 \cdot a_8 \cdot a_5 + 2 \cdot a_9^4 \cdot a_8 \\
& + 2 \cdot a_9^4 \cdot a_7^2 - 2 \cdot a_9^4 \cdot a_7 \cdot a_5 + 12 \cdot a_9^4 \cdot a_7 + a_9^4 \cdot a_6^2 - 10 \cdot a_9^4 \cdot a_6 + 2 \cdot a_9^4 \cdot a_5 - 3 \cdot a_9^4 \\
& + 20 \cdot a_9^3 \cdot a_8^2 - 16 \cdot a_9^3 \cdot a_8 \cdot a_7 - 4 \cdot a_9^3 \cdot a_8 \cdot a_6 - 16 \cdot a_9^3 \cdot a_8 \cdot a_5 + 16 \cdot a_9^3 \cdot a_8 \\
& - 6 \cdot a_9^3 \cdot a_7^2 + 8 \cdot a_9^3 \cdot a_7 - 12 \cdot a_9^3 \cdot a_6 + 2 \cdot a_9^3 \cdot a_5^2 + 4 \cdot a_9^3 - 20 \cdot a_9^2 \cdot a_8^3 \\
& - 20 \cdot a_9^2 \cdot a_8^2 \cdot a_7 + 8 \cdot a_9^2 \cdot a_8^2 \cdot a_6 + 10 \cdot a_9^2 \cdot a_8^2 \cdot a_5 - 8 \cdot a_9^2 \cdot a_8^2 - 8 \cdot a_9^2 \cdot a_8 \cdot a_7^2 \\
& + 4 \cdot a_9^2 \cdot a_8 \cdot a_7 \cdot a_6 + 8 \cdot a_9^2 \cdot a_8 \cdot a_7 \cdot a_5 - 32 \cdot a_9^2 \cdot a_8 \cdot a_7 - 4 \cdot a_9^2 \cdot a_8 \cdot a_6^2 \\
& - 2 \cdot a_9^2 \cdot a_8 \cdot a_6 \cdot a_5 + 32 \cdot a_9^2 \cdot a_8 \cdot a_6 + 4 \cdot a_9^2 \cdot a_8 - 2 \cdot a_9^2 \cdot a_7^3 + a_9^2 \cdot a_7^2 \cdot a_5 \\
& + 12 \cdot a_9^2 \cdot a_7 \cdot a_6 + 16 \cdot a_9^2 \cdot a_7 \cdot a_5 - 4 \cdot a_9^2 \cdot a_7 - 4 \cdot a_9^2 \cdot a_6^2 - 6 \cdot a_9^2 \cdot a_6 \cdot a_5 \\
& + 4 \cdot a_9^2 \cdot a_5^2 + 2 \cdot a_9^2 \cdot a_5 + 8 \cdot a_9^2 - 8 \cdot a_9 \cdot a_8^3 + 16 \cdot a_9 \cdot a_8^2 \cdot a_7 + 8 \cdot a_9 \cdot a_8^2 \cdot a_6 \\
& + 12 \cdot a_9 \cdot a_8^2 \cdot a_5 - 16 \cdot a_9 \cdot a_8^2 + 12 \cdot a_9 \cdot a_8 \cdot a_7^2 - 16 \cdot a_9 \cdot a_8 \cdot a_7 - 8 \cdot a_9 \cdot a_8 \cdot a_6 \cdot a_5 \\
& + 16 \cdot a_9 \cdot a_8 \cdot a_6 - 4 \cdot a_9 \cdot a_8 \cdot a_5^2 - 8 \cdot a_9 \cdot a_8 - 4 \cdot a_9 \cdot a_7^2 \cdot a_5 + 4 \cdot a_9 \cdot a_7^2 \\
& + 8 \cdot a_9 \cdot a_7 \cdot a_6^2 - 16 \cdot a_9 \cdot a_7 \cdot a_6 - 2 \cdot a_9 \cdot a_7 \cdot a_5^2 + 8 \cdot a_9 \cdot a_7 \cdot a_5 - 8 \cdot a_9 \cdot a_6^2 \\
& - 16 \cdot a_9 \cdot a_6 \cdot a_5 + 8 \cdot a_9 \cdot a_6 + 2 \cdot a_9 \cdot a_5^2 + 4 \cdot a_9 \cdot a_5 + 4 \cdot a_8^4 + 8 \cdot a_8^3 \cdot a_7 - 8 \cdot a_8^3 \cdot a_6 \\
& - 4 \cdot a_8^3 \cdot a_5 + 8 \cdot a_8^3 + 8 \cdot a_8^2 \cdot a_7^2 - 8 \cdot a_8^2 \cdot a_7 \cdot a_6 - 4 \cdot a_8^2 \cdot a_7 \cdot a_5 + 16 \cdot a_8^2 \cdot a_7 \\
& + 4 \cdot a_8^2 \cdot a_6^2 + 4 \cdot a_8^2 \cdot a_6 \cdot a_5 - 16 \cdot a_8^2 \cdot a_6 + a_8^2 \cdot a_5^2 - 8 \cdot a_8^2 \cdot a_5 + 4 \cdot a_8 \cdot a_7^3 \\
& - 4 \cdot a_8 \cdot a_7^2 \cdot a_6 - 2 \cdot a_8 \cdot a_7^2 \cdot a_5 - 16 \cdot a_8 \cdot a_7 \cdot a_6 - 16 \cdot a_8 \cdot a_7 \cdot a_5 + 8 \cdot a_8 \cdot a_7 \\
& + 16 \cdot a_8 \cdot a_6^2 + 8 \cdot a_8 \cdot a_6 \cdot a_5 - 8 \cdot a_8 \cdot a_6 - 4 \cdot a_8 \cdot a_5 - 8 \cdot a_8 + a_7^4 - 4 \cdot a_7^3 \\
& - 4 \cdot a_7^2 \cdot a_6 - 6 \cdot a_7^2 \cdot a_5 + 8 \cdot a_7^2 + 8 \cdot a_7 \cdot a_6^2 + 8 \cdot a_7 \cdot a_6 \cdot a_5 - 8 \cdot a_7 \cdot a_6 + 2 \cdot a_7 \cdot a_5^2 \\
& + 4 \cdot a_7 \cdot a_5 - 8 \cdot a_6^3 - 4 \cdot a_6^2 \cdot a_5 + 12 \cdot a_6^2 + 2 \cdot a_6 \cdot a_5^2 + 4 \cdot a_6 \cdot a_5 + a_5^3 + 3 \cdot a_5^2 - 4
\end{aligned}$$

A single run with these parameters took around 10 seconds. The size of the matrix, determined by the number of monomials involved, is  $408 \times 408$ .

This relation poses some interesting questions; since the equation is of degree at least 3 in each of the variables, in general there will not be a *unique* solution for the variable  $a_5$ , given values for the  $a_9, \dots, a_6$ . Moreover, the polynomial is irreducible when we consider it as a polynomial in  $F[a_i]$  for all  $i \in \{9, 8, 7, 6, 5\}$ , with  $F$  the field of rational functions in the other four variables.



Using this — impractical — relation, an improvement factor of  $10/4$  is achieved; less than in XTR but more than in LUC.

*Example 11.*  $k = 12, 24$

For  $k = 12$  and  $d = 1$  we found polynomials  $Q_7$  and  $Q_6$  expressing  $a_7$  and  $a_6$  in  $a_{11}, \dots, a_8$ ; these polynomials are of weighted degree 15 and 18 respectively. As in the previous case they contain powers of  $a_7$  and  $a_6$  greater than 1.

For  $k = 24$ ,  $d = 2$  we found the same relations as for  $k = 12$ ,  $d = 1$ .

*Example 12.*  $k = 30$

Finally, the most interesting case.

$k$	$d$	$e$	$\lceil \phi(k)/d \rceil \cdot d$	Prop. 1
30	1	30	8	15
30	2	15	8	14
30	3	10	9	15
30	5	6	10	10
30	6	5	12	12
30	10	3	10	10
30	15	2	15	15

As before we compare the conjectured and proven bounds on the number of elements of  $\mathbb{F}_p$  that suffices. This shows that three cases of Conjecture 1 are still open. A quick run of Algorithm 2 showed that Conjecture 2 is false in each of the three cases  $(3, 10)$ ,  $(2, 15)$  and  $(1, 30)$ .

The table also shows that to prove Conjecture 3 for  $k = 30$  we either need to prove that 8 elements of  $\mathbb{F}_p$  or 4 elements of  $\mathbb{F}_{p^2}$  will suffice to generate all coefficients. Our further search for relations in these cases had no success either; the search bounds are given below.

$p$	$k$	$d$	$u$	$w$	$\#S$
1009	30	1	8	27	10269
1009	30	1	11	24	6720
1009	30	1	14	25	9012
71	30	2	4	10	3616
71	30	2	5	6	1920
101	30	2	6	7	5760

The last column lists the number of monomials taken into consideration and hence the number of minimal polynomials generated. Note that these results (for  $d = 2$ ) refer to a modification of Algorithm 2, discussed in Section 6, to include conjugates of the coefficients.

For the remaining open cases (see the table below) we searched for relations in vain. For each line in the table we ran Algorithm 1 for every  $u$  in the range from the conjectured value (inclusive) up to the proven bound (exclusive). For values of  $k$  exceeding 20 we ran Algorithm 1 only with  $w = u + 1$  (thus only

testing Conjecture [2], while for  $k \leq 20$  we went further (in an attempt to prove Conjecture [1]), by taking  $w = 2(u + 1)$  or even  $w = 3(u + 1)$ .

When  $d > 1$  we also ran the modified algorithm, taking the conjugates into account; only in the cases  $k = 21$ ,  $d = 3$ ,  $u = 5$ , and  $k = 27$ ,  $d = 3$ ,  $u = 6$ , the resulting computation involved square matrices that were too large for us to deal with. The conjectured improvement factors in both cases are less than 2.

The table lists all 25 cases with  $de \leq 30$  for which Proposition [1] and Example [5] do *not* provide a proof of Conjecture [2]. It lists the value  $\lceil \phi(k)/d \rceil$  for  $u_d^*$  predicted by Conjecture [2] the correct value as obtained by our experiments, and the upper bound  $S/d$  implied by Proposition [1] (cf. Table 1).

$k$	$(d, e)$	$\lceil \phi(k)/d \rceil$	$u_d^*$	$S/d$	$k$	$(d, e)$	$\lceil \phi(k)/d \rceil$	$u_d^*$	$S/d$
9	(1, 9)	6	8	8	24	(1, 24)	8	12	12
10	(1, 10)	4	5	5	24	(2, 12)	4	6	6
12	(1, 12)	4	6	6	24	(3, 8)	3	4	4
14	(1, 14)	6	7	7	25	(1, 25)	20	24	24
15	(1, 15)	8	14	14	26	(1, 26)	12	13	13
15	(3, 5)	3	4	4	27	(1, 27)	18	26	26
18	(1, 18)	6	9	9	27	(3, 9)	6	6, 7, 8	8
18	(2, 18)	3	4	4	28	(1, 28)	12	14	14
20	(1, 20)	8	10	10	28	(2, 14)	6	7	7
20	(2, 10)	4	5	5	30	(1, 30)	8	15	15
21	(1, 21)	12	20	20	30	(2, 15)	4	7	7
21	(3, 7)	4	5, 6	6	30	(3, 10)	3	5	5
22	(1, 22)	10	11	11					

**Theorem 2.** *Conjecture [2] is false for all pairs  $(d, e)$  covered by the table, with the possible exception of the case  $(3, 9)$ . For all  $(d, e)$  with  $de \leq 30$ , with  $(3, 7)$  and  $(3, 9)$  possibly excepted, the true value of  $u_d^*$  equals the upper bound implied by Proposition [1]. Moreover, Conjecture [4] is false for  $k = 30, 21, 15$ , i.e. the cases  $\leq 30$  not covered by Proposition [1].*

## 8 Conclusion

Based on generalisations of the LUC and XTR methods we have formulated precise and verifiable versions of the Brouwer-Pellikaan-Verheul conjecture posed in [4]. By experiment we have shown that it is unlikely that a compact representation of elements exists in extension fields of degree thirty, providing some evidence that XTR cannot be improved with respect to compactness of representation.

Our experiments leave open the possibility that the conjectures hold with polynomials of large degree, which most likely would be of no practical value.

## References

1. M. Adleman, J. DeMarrais *A subexponential algorithm over all finite fields*, CRYPTO '93 Proc., Springer-Verlag, 147-158.
2. D. Bleichenbacher, W. Bosma, A.K. Lenstra, *Some remarks on Lucas-Based Cryptosystems*, CRYPTO '95 Proceedings, Springer-Verlag, pp. 386-396.
3. W. Bosma, J.J. Cannon, C. Playoust, *The Magma Algebra System I: The User Language*, Journal of Symbolic Computation **24** (1997), 235-265.
4. A.E. Brouwer, R. Pellikaan, E.R. Verheul, *Doing more with fewer bits*, Proceedings Asiacrypt99, LNCS 1716, Springer-Verlag 1999, 321-332.
5. D. Cox, J. Little, D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer, 1992.
6. W. Diffie, M.E. Hellman, *New directions in cryptography*, IEEE Trans. on IT **22**, 1976, 644-654.
7. G. Gong, L. Harn, *Public key cryptosystems based on cubic finite field extensions*, IEEE Trans. on I.T., November 1999.
8. S. Lang, *Algebra*, Addison-Welsey, 1993.
9. A.K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields*, Information Security and Privacy - ACISP97 Proceedings (Sydney 1997), Lect. Notes in Comp. Sci. 1270, Springer-Verlag, pp. 127-138.
10. A.K. Lenstra, E.R. Verheul, *The XTR public key system*, Proceedings of Crypto 2000, LNCS 1880, Springer-Verlag, 2000, 1-19; available from [www.ecstr.com](http://www.ecstr.com).
11. A.K. Lenstra, E.R. Verheul, *Key improvements to XTR*, Proceedings of Asiacrypt 2000, LNCS 1976, Springer-Verlag, 2000, 220-223; available from [www.ecstr.com](http://www.ecstr.com).
12. A.K. Lenstra, E.R. Verheul, *Fast irreducibility and subgroup membership testing in XTR*, Proceedings of the 2001 Public Key Cryptography conference, LNCS 1992, Springer-Verlag, 2001, 73-86; available from [www.ecstr.com](http://www.ecstr.com).
13. A.K. Lenstra, E.R. Verheul, *An overview of the XTR public key system*, In: Public-Key Cryptography and Computational Number Theory, Walter de Gruyter, 2001, 151-180.
14. R. Lidl, W.B. Müller, *Permutation Polynomials in RSA-cryptosystems*, Crypto '83 Proceedings, Plenum Press, pp. 293-301.
15. R. Lidl, H. Niederreiter, *Finite Fields*, Addison-Wesley, 1983.
16. W.B. Müller, *Polynomial functions in modern cryptology*, Contributions to general Algebra 3, Proceedings of the Vienna Conference (1985), pp. 7-32. Proceedings, Springer-Verlag, pp. 50-61.
17. W.B. Müller, W. Nöbauer, *Cryptanalysis of the Dickson-Scheme*, Eurocrypt '85 Proceedings, Springer-Verlag, pp. 50-61.
18. W.K. Nicholson, *Introduction to abstract algebra*, PWS-Kent Publishing Company, Boston, 1993.
19. W. Nöbauer, *Cryptanalysis of the Rédei Scheme*, Contributions to general Algebra 3, Proceedings of the Vienna Conference (1985), pp. 255-264.
20. J.M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. Comp., **32** (1978), 918-924.
21. C.P. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, **4** (1991), 161-174.
22. P. Smith, C. Skinner, *A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms*, Asiacrypt '94 proceedings, Springer-Verlag, pp. 357-364.
23. M. Stam, A.K. Lenstra, *Speeding Up XTR*, Proceedings of Asiacrypt 2001, LNCS 2248, Springer-Verlag, 2001, 125-143; available from [www.ecstr.com](http://www.ecstr.com).

# Bounds for Robust Metering Schemes and Their Relationship with $A^2$ -code

Wakaha Ogata<sup>1</sup> and Kaoru Kurosawa<sup>2</sup>

<sup>1</sup> Tokyo Institute of Technology,  
2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan

<sup>2</sup> Ibaraki University,  
4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan

**Abstract.** A metering scheme allows a correct counting on the number of hits that a Web site received during a certain period. In this paper, we first derive tight lower bounds on the communication complexity  $|V_i|$  ( $i = 1, \dots, n$ ) and the size of server's secrets  $|E_s|$  for robust and perfect  $(k, n)$ -metering schemes. We next show an almost equivalence between  $(k, n)$ -metering schemes and  $k$ -multiple-use  $A^2$ -codes. Finally, by using this equivalence, we derive lower bounds on  $|V_i|$  and  $|E_s|$  for robust (but not necessarily perfect)  $(k, n)$ -metering schemes.

## 1 Introduction

A  $(k, n)$ -metering scheme allows a correct counting on the number of hits that a Web site received during a certain period. That is, a Web server  $\mathcal{S}$  can compute a *proof* if and only if  $k$  or more clients visited  $\mathcal{S}$  during a certain period. Naor and Pinkas proposed the first cryptographically secure  $(k, n)$ -metering scheme [1]. Ogata and Kurosawa showed that their scheme is not as secure as they claimed and presented a more secure scheme [2].

More specifically, there exist four kinds of participants, a Web server  $\mathcal{S}$ ,  $n$  clients  $\mathcal{C}_1, \dots, \mathcal{C}_n$ , an audit agency  $\mathcal{A}$  and an outside enemy  $\mathcal{E}$  in this model. (clients are monitors and the outside enemy is not.) We then require the following three kinds of security.

**Security against Servers.** A malicious Web server  $\mathcal{S}$  tries to forge a *proof* from only  $k - 1$  or less shares (authenticators) of clients and to cheat  $\mathcal{A}$ . Hence  $\mathcal{S}$  should not be able to inflate her hit counts. (There appears to be no way to detect whether  $\mathcal{S}$  is deflating her hit counts.)

**Security against Clients.** Malicious clients try to forge an illegal share which would be accepted by  $\mathcal{S}$ , but would not allow  $\mathcal{S}$  to compute the correct *proof*. Hence  $\mathcal{S}$  must be able to detect illegal shares forged by clients.

**Security against Outside Enemy.** An outside enemy  $\mathcal{E}$  tries to forge a (legal or illegal) share which would be accepted by  $\mathcal{S}$ . If it is legal, it causes a counting error because he is not a monitor. If it is illegal, it does not allow  $\mathcal{S}$  to compute the correct *proof*. Hence  $\mathcal{S}$  must be able to detect a share forged by  $\mathcal{E}$ .

We say that a  $(k, n)$ -metering scheme is

- *robust* if it satisfies all the three security requirements.
- *non-robust* if it satisfies only the security against servers.

We further say that a  $(k, n)$ -metering scheme is perfect if  $\mathcal{S}$  gains no information on *proof* from any  $k-1$  or less shares. (It is interesting that the metering schemes proposed so far are all perfect.)

For *non-robust* and perfect metering schemes, a lower bound on the communication complexity  $|V_i|$  ( $i = 1, \dots, n$ ) was shown by De Bonis, B. Masucci [4] and by Masucci and Stinson [3], where  $V_i$  is a set of possible values  $v_i$  which is sent by client  $\mathcal{C}_i$  to  $\mathcal{S}$  when  $\mathcal{C}_i$  has access to  $\mathcal{S}$ . (They considered a more general model than ours such that there are multiple Web servers and there exists a ramp structure among clients.)

However, *non-robust* metering schemes are not practical. We cannot assume that clients are all honest. We cannot assume that there is no outside enemy, either.

In this paper, we derive lower bounds on the communication complexity  $|V_i|$  ( $i = 1, \dots, n$ ) and the size of server's secrets  $|E_s|$  for *robust*  $(k, n)$ -metering schemes.

We first derive lower bounds on  $|V_i|$  and  $|E_s|$  for “perfect and robust”  $(k, n)$ -metering schemes by using counting arguments. We also present a slightly modified version of the Ogata-Kurosawa scheme [2] and prove that it satisfies all the equalities of our bounds. This means that our bounds are all tight.

We next show an almost equivalence between robust  $(k, n)$ -metering schemes and  $k$ -multiple-use A<sup>2</sup>-codes such that we can always construct a  $k$ -multiple-use A<sup>2</sup>-code from a  $(k, n)$ -metering scheme, and in some cases, we can do the reverse. By using this equivalence, we derive lower bounds on  $|V_i|$  and  $|E_s|$  for robust (but not necessarily perfect)  $(k, n)$ -metering schemes. This equivalence is of independent interest because no relationship has been known between them so far.

	Lower bound on $ V_i $	Lower bound on $ E_s $
Non-robust and perfect	[4, 3]	Meaningless*
Robust and perfect	This paper	This paper
Robust	This paper	This paper

(For \*, see the last paragraph of Sec.2.5.)

## 2 Preliminaries

### 2.1 Model of Metering Schemes

A  $(k, n)$ -metering scheme consists of three phases.

**Initialization Phase:** An audit agency  $\mathcal{A}$  first generates a *proof*, a secret key  $e_s$  of the Web server  $\mathcal{S}$  and a share  $v_i$  of client  $\mathcal{C}_i$  for  $i = 1, \dots, n$ .  $\mathcal{A}$  then gives  $e_s$  to  $\mathcal{S}$  and  $v_i$  to  $\mathcal{C}_i$  for  $i = 1, \dots, n$  secretly.

**Communication Phase:** If  $\mathcal{C}_i$  wants to see the Web page of  $\mathcal{S}$ , he sends  $v_i$  to  $\mathcal{S}$ .  $\mathcal{S}$  accepts  $(i, v_i)$  iff  $e_s(i, v_i) = 1$ .

**Proof Computing Phase:** If  $k$  or more clients visited  $\mathcal{S}$  during a certain period, then  $\mathcal{S}$  can compute the *proof* from the  $k$  shares she received.

Let  $Proof$ ,  $E_s$  and  $V_i$  be sets of possible values of the proof, server's key and  $\mathcal{C}_i$ 's share. It is desirable that  $|E_s|$  and  $|V_i|$  are small. Let  $\widehat{Proof}$ ,  $\widehat{E}_s$  and  $\widehat{V}_i$  be the random variables distributed on  $Proof$ ,  $E_s$  and  $V_i$ .

$(k, n)$ -metering schemes must satisfy the security against malicious servers, the security against malicious clients and the security against outside enemies. These security are defined in the following subsections.

## 2.2 Security against Malicious Servers

A  $(k, n)$ -metering scheme must be secure at least against malicious servers. A malicious server tries to forge a *proof* from only  $k - 1$  shares of clients. Hence  $\mathcal{S}$  should not be able to inflate her hit counts. (There appears to be no way to detect whether  $\mathcal{S}$  is deflating her hit counts.)

Formally, a malicious  $\mathcal{S}$  corrupts some  $k - 1$  clients  $\mathcal{C}_{i_1}, \dots, \mathcal{C}_{i_{k-1}}$  adaptively and then obtains their  $k - 1$  shares.  $\mathcal{S}$  next forges a *proof'*, hoping that *proof'* = *proof*. The cheating probability of this attack is defined by

$$P_S \triangleq \max_{i_1, \dots, i_{k-1}} \max_{proof'} \Pr(\widehat{Proof} = proof').$$

It is required that  $P_S$  is negligible in any metering scheme.

## 2.3 Perfect Metering Scheme

We say that a metering scheme is perfect if  $\mathcal{S}$  gains no information on *proof* from any  $k - 1$  shares. Note that this is a stronger notion of security against server's attack than saying only that  $P_S$  is negligible.

**Definition 1.** We say that a  $(k, n)$ -metering scheme is perfect if

$$\Pr(\widehat{Proof} = proof \mid \widehat{E}_s = e_s, \widehat{V}_{i_1} = v_{i_1}, \dots, \widehat{V}_{i_{k-1}} = v_{i_{k-1}}) = \Pr(\widehat{Proof} = proof) \quad (1)$$

for any  $e_s, v_{i_1}, \dots, v_{i_{k-1}}$  and *proof*.

It is interesting that the metering schemes proposed so far are all perfect.

## 2.4 Robust Metering Scheme

We say that a metering scheme is robust if it is secure against malicious clients and outside enemies as well as malicious servers.

Malicious clients try to forge an illegal share which would be accepted by  $\mathcal{S}$ , but would not allow  $\mathcal{S}$  to compute the correct *proof*. An outside enemy tries to forge a (legal or illegal) share which would be accepted by  $\mathcal{S}$ . If it is legal, it causes a counting error because he is not a monitor. If it is illegal, it does not allow  $\mathcal{S}$  to compute the correct *proof*.

**Clients' Attack:** Some (even all) clients collude and make a forged share  $v'_i \neq v_i$  for some client  $\mathcal{C}_i$ . This attack will prevent  $\mathcal{S}$  from computing the *proof* even if  $k$  or more clients visited  $\mathcal{S}$ . (For example, one illegal share and  $k - 1$  honest shares yield an illegal *proof* that is rejected by  $\mathcal{A}$ .) The cheating probability is defined by

$$P_C \triangleq \max_{v_1, \dots, v_n} \max_i \max_{v' \neq v_i} \Pr(\mathcal{S} \text{ accepts } (i, v') \mid v_1, \dots, v_n \text{ are given}).$$

**Outside Enemy's Attack:** An outside enemy is interested in his attack before  $\mathcal{S}$  computes a proof. Therefore, it must send the forged share to  $\mathcal{S}$  before  $\mathcal{S}$  receives  $k$  shares. In other words, the outside enemy can observe at most  $k - 1$  shares sent by clients before computing a forged share. To summarize, the outside enemy makes a forged share  $v'_i$  for some client  $\mathcal{C}_i$  by observing  $l < k$  shares of the other clients. The cheating probability of this attack is defined by

$$P_E \triangleq \max_{0 \leq l < k} \max_{i_1, \dots, i_l} \max_{v_{i_1}, \dots, v_{i_l}} \max_{i \notin \{i_1, \dots, i_l\}, v'} \Pr(\mathcal{S} \text{ accepts } (i, v') \mid \mathcal{E} \text{ observes } v_{i_1}, \dots, v_{i_l}).$$

A metering scheme is called robust if  $P_C$  and  $P_E$  are negligible.

## 2.5 Bounds for Non-robust Metering Scheme

A lower bound on the size of  $|V_i|$  for non-robust and perfect metering schemes was shown by De Bonis, B. Masucci [4] and by Masucci and Stinson [3]. They considered a more general model than ours such that there are multiple servers.

**Proposition 1.** [3, Corollary 3.9] *In a non-robust and perfect  $(k, n)$ -metering scheme for multi servers,*

$$\log_2 |V_i| \geq H(\hat{V}_i) \geq sH(\widehat{Proof})$$

where  $s$  is the number of corrupted servers.

They also generalized their bound to ramp structures among clients.

In non-robust metering schemes,  $\mathcal{S}$  does not need to have any  $e_s \in E_s$  to check the shares of clients because there exist no malicious clients and outside enemies. Therefore, a lower bound on  $|E_s|$  is meaningless in this case.

## 3 Bounds for “Perfect and Robust” Metering Scheme

Non-robust metering schemes are not practical. We cannot assume that clients are all honest. We cannot assume that there is no outside enemy, either.

In this section, we derive a lower bound on  $|V_i|$  and a lower bound on  $|E_s|$  for *perfect and robust*  $(k, n)$ -metering schemes. We also present a slightly modified version of the Ogata-Kurosawa scheme [2] and prove that it satisfies all the equalities of our bounds. This means that our bounds are all tight.

### 3.1 Lower Bound on $|V_i|$

Fix  $i_1, \dots, i_k$  arbitrarily. For each  $1 \leq l \leq k$ , define

$$\begin{aligned} V_{i_l}(e_s, v_{i_1}, \dots, v_{i_{l-1}}) &\triangleq \{v_{i_l} \mid \Pr(\hat{E}_s = e_s, \hat{V}_{i_1} = v_{i_1}, \dots, \hat{V}_{i_{l-1}} = v_{i_{l-1}}, \hat{V}_{i_l} = v_{i_l}) > 0\}, \\ V_{i_l}(v_{i_1}, \dots, v_{i_{l-1}}) &\triangleq \{v_{i_l} \mid \Pr(\hat{V}_{i_1} = v_{i_1}, \dots, \hat{V}_{i_{l-1}} = v_{i_{l-1}}, \hat{V}_{i_l} = v_{i_l}) > 0\}, \\ E_s(v_{i_1}, \dots, v_{i_l}) &\triangleq \{e_s \mid \Pr(\hat{E}_s = e_s, \hat{V}_{i_1} = v_{i_1}, \dots, \hat{V}_{i_l} = v_{i_l}) > 0\}. \end{aligned}$$

Note that  $V_{i_k} \supseteq V_{i_k}(e_s) \supseteq \dots \supseteq V_{i_k}(e_s, v_{i_1}, \dots, v_{i_{k-1}})$ .

**Lemma 1.** *For any possible  $e_s, v_{i_1}, \dots, v_{i_{k-1}}$ ,*

$$|V_{i_k}(e_s, v_{i_1}, \dots, v_{i_{k-1}})| \geq |\text{Proof}|.$$

*Proof.* Fix any possible  $e_s, v_{i_1}, \dots, v_{i_{k-1}}$  arbitrarily. Then any  $\text{proof} \in \text{Proof}$  can happen with positive probability in a perfect  $(k, n)$ -metering scheme. On the other hand, each  $v_{i_k} \in V_{i_k}(e_s, v_{i_1}, \dots, v_{i_{k-1}})$  must determine  $\text{proof} \in \text{Proof}$  uniquely. This means that there exists an onto mapping from  $V_{i_k}(e_s, v_{i_1}, \dots, v_{i_{k-1}})$  to  $\text{Proof}$ . Therefore,

$$|V_{i_k}(e_s, v_{i_1}, \dots, v_{i_{k-1}})| \geq |\text{Proof}|.$$

□

**Corollary 1.**  $|V_i(e_s)| \geq |\text{Proof}|$  for any  $i$ .

**Theorem 1.** *In a perfect and robust  $(k, n)$ -metering scheme,*

$$|V_i| \geq |\text{Proof}|(P_E)^{-1}$$

for any  $i$ .

*Proof.* We will derive a lower bound on  $P_E$ . Define

$$\phi(e_s, v_i) \triangleq \begin{cases} 1 & \text{if } v_i \in V_i(e_s) \\ 0 & \text{otherwise.} \end{cases}$$

Note that  $\mathcal{S}$  accepts  $(i, v_i)$  iff  $v_i \in V_i(e_s)$ . Therefore,

$$\begin{aligned} \sum_{v_i \in V_i} \Pr(\mathcal{S} \text{ accepts } (i, v_i)) &= \sum_{v_i \in V_i} \sum_{e_s \in E_s} \Pr(e_s) \phi(e_s, v_i) \\ &= \sum_{e_s \in E_s} \Pr(e_s) \sum_{v_i \in V_i} \phi(e_s, v_i) \\ &= \sum_{e_s \in E_s} \Pr(e_s) |V_i(e_s)| \\ &\geq |\text{Proof}| \sum_{e_s \in E_s} \Pr(e_s) && (\text{from Corollary 1}) \\ &= |\text{Proof}|. \end{aligned}$$

Therefore,

$$P_E \geq \max_{v_i \in V_i} \Pr(\mathcal{S} \text{ accepts } (i, v_i)) \geq |\text{Proof}|/|V_i|.$$

□



### 3.2 Lower Bound on $|E_s|$

Define

$$ALL \triangleq \{(v_1, \dots, v_k) \mid \Pr(\hat{V}_1 = v_1, \dots, \hat{V}_k = v_k) > 0\},$$

$$ALL(e_s) \triangleq \{(v_1, \dots, v_k) \mid \Pr(\hat{E}_s = e_s, \hat{V}_1 = v_1, \dots, \hat{V}_k = v_k) > 0\}.$$

**Lemma 2.** *If the equality of corollary 1 holds for all  $i$ , then*

$$|ALL(e_s)| = |Proof|^k.$$

*Proof.* From the equality of Corollary 1 and Lemma 1,

$$|Proof| = |V_2(e_s)| \geq |V_2(e_s, v_1)| \geq |Proof|.$$

Therefore,  $|V_2(e_s, v_1)| = |Proof|$  for any  $v_1 \in V_1(e_s)$ . Hence,

$$|\{(v_1, v_2) \mid \Pr(\hat{E}_s = e_s, \hat{V}_1 = v_1, \hat{V}_2 = v_2) > 0\}| = |V_1(e_s)| \times |V_2(e_s, v_1)| = |Proof|^2.$$

By repeating this process, we have  $|ALL(e_s)| = |Proof|^k$ .  $\square$

**Lemma 3.**  $|V_{i+1}(v_{i_1}, \dots, v_{i_l})| \geq |Proof|(P_E)^{-1}$  for  $1 \leq l \leq k-1$ .

*Proof.* Similar to the proof of Theorem 1. Suppose that an outside enemy  $\mathcal{E}$  observes  $l$  shares sent by clients, say  $\mathcal{C}_{i_1}, \dots, \mathcal{C}_{i_l}$ . Let their shares be  $\mathbf{v} = (v_{i_1}, \dots, v_{i_l})$ . Define

$$\phi(e_s, v_{i_{l+1}}) \triangleq \begin{cases} 1 & \text{if } v_{i_{l+1}} \in V_{i_{l+1}}(e_s) \\ 0 & \text{otherwise.} \end{cases}$$

Note that  $\mathcal{S}$  accepts  $(i_{l+1}, v_{i_{l+1}})$  iff  $v_{i_{l+1}} \in V_{i_{l+1}}(e_s)$ . Therefore,

$$\begin{aligned} & \sum_{v_{i_{l+1}} \in V_{i_{l+1}}(\mathbf{v})} \Pr(\mathcal{S} \text{ accepts } (i_{l+1}, v_{i_{l+1}}) \mid \mathcal{E} \text{ observes } \mathbf{v}) \\ &= \sum_{v_{i_{l+1}} \in V_{i_{l+1}}(\mathbf{v})} \sum_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}) \phi(e_s, v_{i_{l+1}}) \\ &= \sum_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}) \sum_{v_{i_{l+1}} \in V_{i_{l+1}}(\mathbf{v})} \phi(e_s, v_{i_{l+1}}) \\ &= \sum_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}) |V_{i_{l+1}}(\mathbf{v}) \cap V_{i_{l+1}}(e_s)| \\ &= \sum_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}) |V_{i_{l+1}}(e_s, \mathbf{v})| \\ &\geq |Proof| \sum_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}) \quad (\text{from Lemma 1}) \\ &= |Proof|. \end{aligned}$$

Therefore,

$$P_E \geq |Proof|/|V_{i_{l+1}}(\mathbf{v})|.$$

$\square$

**Lemma 4.**  $|E_s(v_1, \dots, v_k)| \geq P_C^{-1}$ .

*Proof.* Consider the following attack of  $k$  clients  $\mathcal{C}_1, \dots, \mathcal{C}_k$ . Let their shares be  $\mathbf{v} = (v_1, \dots, v_k)$ . First, they choose  $e'_s \in E_s(\mathbf{v})$  such that

$$\Pr(e'_s \mid \mathbf{v}) = \max_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}).$$

Next they choose  $v'_1$  randomly from  $V_1(e'_s, v_2, \dots, v_k) \setminus \{v_1\}$ . Finally  $\mathcal{C}_1$  sends  $v'_1$  to  $\mathcal{S}$ . Clearly, this attack succeeds if  $\mathcal{S}$  has  $e'_s$ . Therefore,

$$P_C \geq \Pr(\mathcal{S} \text{ has } e'_s \mid \mathbf{v}) = \max_{e_s \in E_s(\mathbf{v})} \Pr(e_s \mid \mathbf{v}) \geq 1/|E_s(\mathbf{v})|.$$

□

**Lemma 5.**  $|ALL| \geq |Proof|^k (P_E^k)^{-1}$ .

*Proof.* First from Theorem 1,

$$|V_1| \geq |Proof|(P_E)^{-1}.$$

Next from Lemma 3,

$$|V_2(v_1)| \geq |Proof|(P_E)^{-1}$$

for each  $v_1 \in V_1$ . Therefore,

$$|\{(v_1, v_2) \mid \Pr(\hat{V}_1 = v_1, \hat{V}_2 = v_2) > 0\}| \geq |Proof|^2 (P_E^2)^{-1}.$$

By repeating this process, we obtain that  $|ALL| \geq |Proof|^k (P_E^k)^{-1}$ . □

**Theorem 2.** Suppose that the equality of Corollary 1 holds for all  $i$  and  $e_s$ . Then in a perfect and robust  $(k, n)$ -metering scheme,

$$|E_s| \geq (P_C P_E^k)^{-1}.$$

*Proof.* First from Lemma 2,

$$\sum_{e_s \in E_s} |ALL(e_s)| = |E_s| |Proof|^k.$$

Next

$$\sum_{(v_1, \dots, v_k)} |E_s(v_1, \dots, v_k)| \geq |ALL| P_C^{-1} \quad (\text{Lemma 4})$$

$$\geq |Proof|^k (P_E^k)^{-1} (P_C)^{-1} \quad (\text{Lemma 5}).$$

On the other hand, it is easy to see that

$$\sum_{e_s \in E_s} |ALL(e_s)| = \sum_{(v_1, \dots, v_k)} |E_s(v_1, \dots, v_k)|.$$

Therefore,

$$|E_s| \geq \frac{|Proof|^k (P_E^k)^{-1} (P_C)^{-1}}{|Proof|^k} = (P_E^k P_C)^{-1}.$$

□

### 3.3 Modified Ogata-Kurosawa Scheme

We next present a slightly modified version of the Ogata-Kurosawa scheme [2] and prove that it satisfies all the equalities of our bounds. This means that our bounds are all tight.

The modified Ogata-Kurosawa scheme is described as follows. Let  $p > n$  be a large prime number.

- Initialization Phase:**
1. An audit agency  $\mathcal{A}$  chooses a random number  $r \in \mathbb{Z}_p$  and two random polynomials  $f_0(y)$  and  $f_1(y)$  with degree at most  $k - 1$  over  $GF(p)$ .
  2. Let  $proof = f_1(0)$ .
  3.  $\mathcal{A}$  gives  $e_s = (r, g(y))$  to the Web server  $\mathcal{S}$ , where

$$g(y) = f_0(y) + rf_1(y).$$

4.  $\mathcal{A}$  gives  $v_i = (f_0(i), f_1(i))$  to client  $\mathcal{C}_i$  for  $1 \leq i \leq n$ .

**Communication Phase:** If  $\mathcal{C}_i$  wants to see the Web page of  $\mathcal{S}$ , he sends  $v_i = (a, b)$  to  $\mathcal{S}$ .  $\mathcal{S}$  accepts  $(i, (a, b))$  iff

$$g(i) = a + rb. \quad (2)$$

**Proof Computing Phase:** If  $k$  or more clients visited  $\mathcal{S}$ , then  $\mathcal{S}$  can compute  $proof = f_1(0)$  by using Lagrange formula.

In the above scheme, it is clear that

$$|Proof| = p, \quad |E_s| = p^{k+1}, \quad |V_i| = p^2$$

for each  $i$ . We then prove the following theorem.

**Theorem 3.** *The modified Ogata-Kurosawa scheme is perfect and*

$$P_C = P_E = 1/p. \quad (3)$$

*Proof.* Note that the secret key of  $\mathcal{A}$  is  $K = (r, f_0(y), f_1(y))$ .

1. For simplicity, let  $i_1 = 1, \dots, i_{k-1} = k - 1$ . Fix

$$e_s = (r, g(y)), v_1 = (a_1, b_1), \dots, v_{k-1} = (a_{k-1}, b_{k-1})$$

arbitrarily. We will show that there exists a unique  $(f_0(y), f_1(y))$  for each value of  $proof$ . Fix  $proof$  arbitrarily. First there exists a unique  $f_1(y)$  such that

$$f_1(0) = proof, f_1(1) = b_1, \dots, f_1(k-1) = b_{k-1}$$

because  $\deg(f_1)$  is at most  $k - 1$ . Next  $f_0(y)$  is uniquely determined as

$$f_0(y) = g(y) - rf_1(y)$$

because  $e_s = (r, g(y))$  is fixed. Therefore, each value of  $proof$  is equally likely to happen for any fixed  $e_s, v_1, \dots, v_{k-1}$ . This means that

$$\Pr(\widehat{Proof} = proof \mid e_s, v_1, \dots, v_{k-1}) = 1/p = \Pr(\widehat{Proof} = proof).$$

Hence the scheme is perfect.

2. Fix

$$v_1 = (a_1, b_1), \dots, v_n = (a_n, b_n) \quad (4)$$

and  $i \in \{1, \dots, n\}$  arbitrarily. Let  $B_0$  be the set of  $K = (r, f_0(y), f_1(y))$  such that eq.(4) holds. For  $v' = (a', b')$  such that  $(a', b') \neq (a_i, b_i)$ , let  $B_1$  be a subset of  $B_0$  such that  $\mathcal{S}$  accepts  $(i, v')$ . Then

$$P_C = \max \Pr(\mathcal{S} \text{ accepts } (i, v') \mid v_1, \dots, v_n) = \max |B_1|/|B_0|.$$

We will compute  $|B_0|$  and  $|B_1|$ . First since  $f_0(y)$  and  $f_1(y)$  are uniquely determined from eq.(4), we have

$$|B_0| = |\{r\}| = p.$$

Next since  $\mathcal{S}$  accepts  $(i, v')$ ,  $g(i) = a' + rb'$ . On the other hand, from eq.(2),  $g(i) = a_i + rb_i$ . Therefore,

$$\begin{aligned} a_i + rb_i &= a' + rb', \\ r(b_i - b') &= a' - a_i. \end{aligned}$$

The above equation has at most one solution on  $r$  because  $(a', b') \neq (a_i, b_i)$ . Therefore,  $\max |B_1| = 1$ . Hence

$$P_C = \max |B_1|/|B_0| = 1/p.$$

3. For simplicity, let  $l = k - 1$  and  $i_1 = 1, \dots, i_{k-1} = k - 1$ . Fix

$$v_1 = (a_1, b_1), \dots, v_{k-1} = (a_{k-1}, b_{k-1}) \quad (5)$$

and  $i(\geq k)$  arbitrarily. Let  $B_0$  be the set of  $K = (r, f_0(y), f_1(y))$  such that eq.(5) holds. For  $v' = (a', b')$  let  $B_1$  be a subset of  $B_0$  such that  $\mathcal{S}$  accepts  $(i, v')$ . Then

$$P_E = \max \Pr(\mathcal{S} \text{ accepts } (i, v') \mid v_1, \dots, v_{k-1}) = \max |B_1|/|B_0|.$$

We will compute  $|B_0|$  and  $|B_1|$ . First since  $f_0(y)$  and  $f_1(y)$  are uniquely determined from the values of  $f_0(0)$  and  $f_1(0)$ , we have

$$|B_0| = |\{r, f_0(0), f_1(0)\}| = p^3.$$

Next  $g(i) = a' + rb'$  if  $\mathcal{S}$  accepts  $(i, v')$ . On the other hand,  $g(i) = f_0(i) + rf_1(i)$ . Therefore,

$$f_0(i) + rf_1(i) = a' + rb'.$$

In the above equation,  $f_0(i)$  is uniquely determined from each values of  $(r, f_1(i))$ . (Note that  $f_0(y)$  and  $f_1(y)$  are uniquely determined from each values of  $f_0(i)$  and  $f_1(i)$ .) Therefore,

$$|B_1| = |\{r, f_1(i)\}| = p^2.$$

Hence

$$P_E = \max |B_1|/|B_0| = 1/p.$$

□

It is now easy to see that all the equalities of our bounds are satisfied by the above scheme.

(Remark) In the original Ogata-Kurosawa scheme,  $proof = f_0(0)$  and  $r$  is randomly chosen from  $Z_p \setminus \{0\}$ . Therefore,  $P_C = 1/(p-1)$  and  $|E_s| = (p-1)p^k$ .

## 4 Lower Bounds for Multiple-Use A<sup>2</sup>-code

For multiple-use A<sup>2</sup>-codes, Wang. et.al. derived a lower bound on the cheating probabilities and a lower bound on the size of keys [8]. (See Appendix A.) However, their bound on the size of keys holds under the condition that the cheating probabilities satisfy their lower bound (see Proposition 3). We can not derive a lower bound on the size of authenticators from their result, either.

In this section, we first define the cheating probabilities in a different way from [8]. We then derive a lower bound on the size of keys which holds for any values of the cheating probabilities. We derive a lower bound on the size of authenticators, also.

The result of this section will be used in the following sections.

### 4.1 Multiple-Use A<sup>2</sup>-code

In the model for unconditionally secure authentication codes (A-codes), the transmitter  $\mathcal{T}$  and the receiver  $\mathcal{R}$  use the same encoding rule to protect their communication from deception of an outside enemy  $\mathcal{O}$ .

An authentication code with arbitration (A<sup>2</sup>-code) enables to authenticate a message sent by  $\mathcal{T}$  to  $\mathcal{R}$  even if  $\mathcal{T}$  and  $\mathcal{R}$  do not trust each other [6, 7]. A<sup>2</sup>-code includes the fourth person called an *arbiter*  $\mathcal{A}'$ , who solves disputes between  $\mathcal{T}$  and  $\mathcal{R}$ .

In this paper, we consider A<sup>2</sup>-codes which are used to send multiple messages. If  $\mathcal{T}$  can use an A<sup>2</sup>-code to send  $k-1$  messages to  $\mathcal{R}$  which are authenticated, then we call the code a *k-multiple-use A<sup>2</sup>-code*.

A *k-multiple-use A<sup>2</sup>-code* consists of three phases.

**Initialization Phase:** An arbiter  $\mathcal{A}'$  first generates a secret key  $e_t$  of  $\mathcal{T}$  and a secret key  $e_r$  of  $\mathcal{R}$ .  $\mathcal{A}'$  then gives  $e_t$  to  $\mathcal{T}$  and  $e_r$  to  $\mathcal{R}$  secretly.

**Communication Phase:** For a source state  $s$ ,  $\mathcal{T}$  computes an authenticator  $a = e_t(s)$ .  $\mathcal{T}$  then sends  $m = (s, a)$  to  $\mathcal{R}$ , where  $m$  is called a message.  $\mathcal{R}$  accepts  $m = (s, a)$  as authentic iff  $e_r(s, a) = 1$ .

**Dispute Phase:** On dispute between  $\mathcal{T}$  and  $\mathcal{R}$ ,  $\mathcal{A}'$  accepts  $m = (s, a)$  as authentic iff  $a = e_t(s)$ .

Define  $E_t \triangleq \{e_t\}$ ,  $E_r \triangleq \{e_r\}$ ,  $M \triangleq \{m\}$ ,  $S \triangleq \{s\}$  and  $A \triangleq \{a\}$ . Let  $\hat{E}_t, \hat{E}_r, \hat{M}, \hat{S}, \hat{A}$  be the random variables distributed over  $E_t, E_r, M, S, A$ , respectively.

In the model of *k-multiple-use A<sup>2</sup>-codes*, there are three kinds of attacks.

**Transmitter's Attack:**  $\mathcal{T}$  sends a message  $m = (s, a)$  to the receiver  $\mathcal{R}$  and denies having sent it.  $\mathcal{T}$  succeeds if  $m$  is accepted by  $\mathcal{R}$  as authentic and  $a \neq e_t(s)$ .

**Receiver's Attack:**  $\mathcal{R}$  receives less than  $k$  messages and claims to have received a new message  $m' = (s', a')$ .  $\mathcal{R}$  succeeds if  $a' = e_t(s')$ .

**Outside Enemy's Attack:** An outside enemy  $\mathcal{O}$  observes  $i < k$  messages sent by  $\mathcal{T}$ , and then substitutes the last one with a forged one  $m' = (s', a')$ .  $\mathcal{O}$  succeeds if  $e_r(s', a') = 1$ .

We define the cheating probabilities of  $k$ -multiple-use A<sup>2</sup>-code as follows, where  $P_T$ ,  $P_{R_i}$  and  $P_{O_i}$  denote the cheating probabilities by  $\mathcal{T}$ ,  $\mathcal{R}$  and  $\mathcal{O}$ , respectively.

$$\begin{aligned} P_T &\triangleq E \left( \max_{m=(s,a), a \neq e_t(s)} \Pr(e_r(s, a) = 1) \right) \\ P_{R_i} &\triangleq E \left( \max_{(s', a') \notin \{m_1, \dots, m_i\}} \Pr(a' = e_t(s') \mid \mathcal{T} \text{ sent } m_1, \dots, m_i) \right) \\ P_{O_i} &\triangleq E \left( \max_{m' \notin \{m_1, \dots, m_i\}} \Pr(\mathcal{R} \text{ accepts } m' \mid \mathcal{T} \text{ sent } m_1, \dots, m_i) \right), \end{aligned}$$

where  $0 \leq i \leq k-1$ . Let

$$P_O \triangleq \max_{0 \leq i < k} P_{O_i}, \quad P_R \triangleq \max_{0 \leq i < k} P_{R_i}.$$

## 4.2 Lower Bounds

In this subsection, we present a lower bound on the cheating probabilities defined as above. It is a generalization of a lower bound for usual A<sup>2</sup>-codes given by Johansson [10].

**Theorem 4.**

$$\begin{aligned} P_T &\geq 2^{-H(\hat{E}_r | \hat{E}_t)} \\ P_{R_i} &\geq 2^{-H(\hat{E}_t | \hat{M}_1 \dots \hat{M}_i, \hat{E}_r) + H(\hat{E}_t | \hat{M}_1 \dots \hat{M}_{i+1}, \hat{E}_r)} \\ P_{O_i} &\geq 2^{-I(\hat{E}_r; \hat{E}_t | \hat{M}_1 \dots \hat{M}_i) + I(\hat{E}_r; \hat{E}_t | \hat{M}_1 \dots \hat{M}_{i+1})} \end{aligned}$$

The proof will be given in the final paper. We then obtain a lower bound on the size of keys as follows.

**Theorem 5.** *If  $\hat{S}$  is uniformly distributed, then*

$$|E_t| \geq (P_R P_O)^{-k}, \quad |E_r| \geq (P_T P_O^k)^{-1}, \quad |A| \geq (P_R P_O)^{-1}.$$

*Proof.* From Theorem 4,

$$(P_R)^k \geq (P_{R_0} \dots P_{R_{k-1}}) \geq 2^{-H(\hat{E}_t | \hat{E}_r) + H(\hat{E}_t | \hat{M}_1 \dots \hat{M}_k, \hat{E}_r)}$$

$$\begin{aligned}
 (P_O)^k &\geq (P_{O_0} \cdots P_{O_{k-1}}) \geq 2^{-I(\hat{E}_t; \hat{E}_r) + I(\hat{E}_t; \hat{E}_r | \hat{M}_1 \cdots \hat{M}_k)} \\
 (P_O P_R)^k &\geq 2^{-H(\hat{E}_t) + H(\hat{E}_t | \hat{M}_1 \cdots \hat{M}_k)} \\
 &\geq 2^{-H(\hat{E}_t)}, \\
 |E_t| &\geq 2^{H(\hat{E}_t)} \geq (P_O P_R)^{-k}.
 \end{aligned}$$

The second bound can be derived similarly.

The bound on  $|A|$  is derived as follows.

$$\begin{aligned}
 |M| &\geq 2^{H(\hat{M} | \hat{E}_r) + I(\hat{M}; \hat{E}_r)} \\
 &= 2^{H(\hat{M} | \hat{E}_r)} 2^{I(\hat{M}; \hat{E}_r; \hat{E}_t)} 2^{I(\hat{M}; \hat{E}_t | \hat{E}_r)} \\
 &\geq 2^{H(\hat{S})} 2^{I(\hat{E}_r; \hat{E}_t) - I(\hat{E}_r; \hat{E}_t | \hat{M})} 2^{H(\hat{E}_t | \hat{E}_r) - H(\hat{E}_t | \hat{M}, \hat{E}_r)}
 \end{aligned}$$

From Theorem 4, it holds that

$$\begin{aligned}
 2^{I(\hat{E}_r; \hat{E}_t) - I(\hat{E}_r; \hat{E}_t | \hat{M})} &\geq 1/P_{O_0}, \\
 2^{H(\hat{E}_t | \hat{E}_r) - H(\hat{E}_t | \hat{M}, \hat{E}_r)} &\geq 1/P_{R_0}.
 \end{aligned}$$

Therefore,

$$\begin{aligned}
 |M| &\geq 2^{H(\hat{S})} / P_{O_0} P_{R_0} = |S| / P_{O_0} P_{R_0}, \\
 |A| &= |M| / |S| \geq (P_O P_R)^{-1}.
 \end{aligned}$$

□

We can see that the above bounds are tight because there exists an A<sup>2</sup>-code which satisfies all the equalities of them (see appendix B).

## 5 Almost Equivalence

In this section, we show an almost equivalence between robust  $(k, n)$ -metering schemes and  $k$ -multiple-use A<sup>2</sup>-codes such that we can always construct a  $k$ -multiple-use A<sup>2</sup>-code from a  $(k, n)$ -metering scheme, and in some cases, we can do the reverse.

In what follows, we define the cheating probability of clients and the cheating probability of outside enemies as follows.

$$\tilde{P}_C \triangleq E \left( \max_i \max_{v'_i \neq v_i} \Pr(\mathcal{S} \text{ accepts } (i, v'_i) \mid v_1, \dots, v_n \text{ are given}) \right),$$

where  $E$  is taken over  $v_1, \dots, v_n$ .

$$\tilde{P}_E \triangleq \max_{0 \leq l < k} E \left( \max_{i \notin \{i_1, \dots, i_l\}, v'_i} \Pr(\mathcal{S} \text{ accepts } (i, v'_i) \mid \mathcal{E} \text{ observes } v_{i_1}, \dots, v_{i_l}) \right),$$

where  $E$  is taken over  $i_1, \dots, i_l$  and  $v_{i_1}, \dots, v_{i_l}$ .

The cheating probabilities of  $k$ -multiple-use A<sup>2</sup>-codes are defined in the previous section.

### 5.1 Metering Scheme Implies a Multiple-Use $A^2$ -code

First, we show that a  $(k, n)$ -metering scheme implies a  $k$ -multiple-use  $A^2$ -code. Wlog, suppose that  $V_i \subseteq V$ , where  $|V| = \max_i |V_i|$ .

**Theorem 6.** *If there exists a  $(k, n)$ -metering scheme with  $(\text{Proof}, E_s, \{V_i\})$  and  $(\tilde{P}_C, P_S, \tilde{P}_E)$ , then there exists a  $k$ -multiple-use  $A^2$ -code with  $(E_t, E_r, S, A)$  and  $(P_T, P_R, P_O)$  such that*

$$\begin{aligned} P_T &= \tilde{P}_C, & P_R &\leq P_S, & P_O &= \tilde{P}_E, \\ E_t &= V_1 \times \cdots \times V_n, & E_r &= E_s, & S &= \{1, 2, \dots, n\}, & A &= V. \end{aligned}$$

*Proof.* Suppose that there exists a  $(k, n)$ -metering scheme with  $(\text{Proof}, E_s, \{V_i\})$  and  $(\tilde{P}_C, P_S, \tilde{P}_E)$ . We then construct a  $k$ -multiple-use  $A^2$ -code as follows.

**Initialization Phase:** The arbiter  $\mathcal{A}'$  first runs the audit agency  $\mathcal{A}$  of the  $(k, n)$ -metering scheme to generate *proof*,  $e_s$  and  $(v_1, \dots, v_n)$ .  $\mathcal{A}'$  then gives

$e_t \triangleq (v_1, \dots, v_n)$  to  $\mathcal{T}$  and  $e_r \triangleq e_s$  to  $\mathcal{R}$  secretly as their secret keys.

**Communication Phase:** For a source state  $i \in \{1, \dots, n\}$ ,  $\mathcal{T}$  sends a message  $m = (i, v_i)$  to  $\mathcal{R}$ , where  $v_i$  is the authenticator for  $i$ .

**Dispute Phase:** On dispute between  $\mathcal{T}$  and  $\mathcal{R}$ ,  $\mathcal{A}'$  accepts  $m = (i, a)$  as authentic iff  $a = v_i$ .

It is clear that  $E_t = V_1 \times \cdots \times V_n, E_r = E_s, S = \{1, 2, \dots, n\}, A = V$ .

Next it is easy to see that an outside enemy's attack on the  $(k, n)$ -metering scheme can be directly used as an outside enemy's attack on the  $k$ -multiple-use  $A^2$ -code and vice versa. Therefore,  $P_O = \tilde{P}_E$ .

A clients' attack on the  $(k, n)$ -metering scheme is that all clients collude and make a forged share  $v'_s \neq v_s$ . In other words, from given  $(1, v_1), \dots, (n, v_n)$ , they make  $v'_s \neq v_s$  for some  $s$ , hoping that it is accepted by  $\mathcal{S}$  with her secret key  $e_s$ . Then it is easy to see that this attack can be directly used as a transmitter's attack on the  $k$ -multiple-use  $A^2$ -code. Therefore,  $P_T \geq \tilde{P}_C$ . It is easy to see that the converse part is also true. Hence  $\tilde{P}_C \geq P_T$ . Therefore,  $P_T = \tilde{P}_C$ .

Suppose that there exists a receiver's attack  $R_{\text{attack}}$  on the  $k$ -multiple-use  $A^2$ -code with success probability  $P_R$ . Then we consider a server's attack on the  $(k, n)$ -metering scheme as follows. Suppose that  $l < k$  clients  $\mathcal{C}_{i_1}, \dots, \mathcal{C}_{i_l}$  visited  $\mathcal{S}$ .  $\mathcal{S}$  runs  $R_{\text{attack}}$  on input  $e_r$  and  $l$  messages  $(i_1, v_{i_1}), \dots, (i_l, v_{i_l})$ .  $R_{\text{attack}}$  outputs a new message  $m = (s, v_s)$  for some  $s \notin \{i_1, \dots, i_l\}$ .  $\mathcal{S}$  next corrupts  $k - l - 1$  clients  $\mathcal{C}_{i_{l+1}}, \dots, \mathcal{C}_{i_{k-1}}$  other than  $\{i_1, \dots, i_l, s\}$  and obtains their shares. Then  $\mathcal{S}$  obtains  $k$  shares  $v_{i_1}, \dots, v_{i_{k-1}}$  and  $v_s$  in total. Therefore,  $\mathcal{S}$  can compute the *proof* from the  $k$  shares. This attack succeeds with probability  $P_R$ . Hence,  $P_S \geq P_R$ .  $\square$

### 5.2 Weak Converse

Next, we show a weak converse of Theorem 6.



**Lemma 6.** *In a  $k$ -multiple-use  $A^2$ -code in which  $|E_t|$  satisfies the equality of the bound in Theorem 5, the transmitter's key is determined uniquely from  $k$  or more valid messages.*

*Proof.* From the proof of Theorem 5, We obtain

$$(P_O P_R)^k \geq 2^{-H(\hat{E}_t) + H(\hat{E}_t | \hat{M}_1 \cdots \hat{M}_k)}.$$

The equality of the bound holds only if  $H(\hat{E}_t | \hat{M}_1 \cdots \hat{M}_k) = 0$ . This means that  $e_t$  is determined by  $k$  messages.  $\square$

**Theorem 7.** *If there exists a  $k$ -multiple-use  $A^2$ -code with  $(E_t, E_r, S, A)$  and  $(P_T, P_R, P_O)$  such that  $|E_t|$  satisfies the equality of the bound in Theorem 5, then there exists a  $(k, n)$ -metering scheme with  $(Proof, E_s, \{V_i\})$  and  $(\tilde{P}_C, P_S, \tilde{P}_E)$ , such that*

$$\begin{aligned} \tilde{P}_C &= P_T, \quad P_S \leq P_R, \quad \tilde{P}_E = P_O \\ E_s &= E_r, \quad n = |S| - 1, \quad Proof = V_1 = \cdots = V_n = A. \end{aligned}$$

*Proof.* (Sketch) Using a  $k$ -multiple-use  $A^2$ -code, construct a metering scheme described as follows.  $\mathcal{A}$  chooses  $s_0 \in S$  and sets  $proof = e_t(s_0)$ . Each client  $\mathcal{C}_i$  receives  $v_i = e_t(s_i)$  where  $S = \{s_0, \dots, s_n\}$ .

If  $|E_t|$  satisfies the equality of the bound,  $e_t$  is determined uniquely from  $k$  or more valid messages (from Lemma 6). Then the server can obtain  $proof = e_t(s_0)$  if he has been visited by  $k$  or more clients. The rest of the proof is similar to Theorem 6.  $\square$

## 6 Lower Bounds for Robust Metering Scheme

In this section, we derive a lower bound on  $|V_i|$  and a lower bound on  $|E_s|$  for robust (but not necessarily perfect)  $(k, n)$ -metering schemes by using our relationship between metering schemes and multiple  $A^2$ -codes (and our lower bounds for  $k$ -multiple-use  $A^2$ -codes of Sec.4).

### 6.1 Bounds for Robust Metering Schemes

From Theorem 5 and Theorem 6, we immediately obtain a lower bound on the size of keys for  $(k, n)$ -metering schemes as follows.

**Corollary 2.** *In a  $(k, n)$ -metering scheme, if each client visits the Web sever  $\mathcal{S}$  with equal probability, then*

$$\max_i |V_i| \geq (P_S \tilde{P}_E)^{-1}, \quad |E_s| \geq (\tilde{P}_C \tilde{P}_E^k)^{-1}.$$

Corollary 2 is tight because the Ogata-Kurosawa metering scheme satisfies all the equalities of the bound (see Sec.3.3).

## 6.2 Bound on $\tilde{P}_E$

We can remove  $\tilde{P}_E$  from the above bound by using Theorem 8.

**Theorem 8.** *In a  $(k, n)$ -metering scheme,*

$$\tilde{P}_E \leq \tilde{P}_C + P_S.$$

*Proof.* (Sketch) From the definition of  $\tilde{P}_E$ ,

$$\begin{aligned} \tilde{P}_E \leq & \max_l E \left( \max_{i, v'_i} \Pr(\mathcal{S} \text{ accepts } (i, v'_i) \wedge \mathcal{C}_i \text{ has } v'_i \mid \mathcal{S} \text{ observes } v_{i_1}, \dots, v_{i_l}) \right) \\ & + \max_l E \left( \max_{i, v'_i} \Pr(\mathcal{S} \text{ accepts } (i, v'_i) \wedge \neg(\mathcal{C}_i \text{ has } v'_i) \mid \mathcal{S} \text{ observes } v_{i_1}, \dots, v_{i_l}) \right). \end{aligned}$$

The first term of the right hand is equal or less than  $P_S$ , while the second term is equal or less than  $\tilde{P}_C$ .  $\square$

**Corollary 3.** *In a  $(k, n)$ -metering scheme, if each client visits the Web sever  $\mathcal{S}$  with equal probability, then*

$$\max_i |V_i| \geq (P_S(P_S + \tilde{P}_C))^{-1}, \quad |E_s| \geq (\tilde{P}_C(P_S + \tilde{P}_C)^k)^{-1}.$$

## 7 Conclusion

In this paper, We first derived lower bounds on  $|V_i|$  and  $|E_s|$  for “perfect and robust”  $(k, n)$ -metering schemes by using counting arguments, where  $|V_i|$  ( $i = 1, \dots, n$ ) is the communication complexity and  $|E_s|$  is the size of server’s secrets. We also presented a slightly modified version of the Ogata-Kurosawa scheme [2] and proved that it satisfies all the equalities of our bounds. This means that our bounds are all tight.

We next showed an almost equivalence between robust  $(k, n)$ -metering schemes and  $k$ -multiple-use  $A^2$ -codes such that we can always construct a  $k$ -multiple-use  $A^2$ -code from a  $(k, n)$ -metering scheme, and in some cases, we can do the reverse. By using this equivalence, we derived lower bounds on  $|V_i|$  and  $|E_s|$  for robust (but not necessarily perfect)  $(k, n)$ -metering schemes. This equivalence is of independent interest because no relationship has been known between them so far.

## References

1. M. Naor and B. Pinkas, “Secure and Efficient Metering,” Proc. of Eurocrypt ’98, Lecture Notes in Computer Science, Vol.1403, pp.576–589 (1998)
2. W. Ogata and K. Kurosawa, “Provably Secure Metering Scheme,” Proc. of Asiacrypt 2000, Lecture Notes in Computer Science, Vol.1976, pp.388–398 (2000)
3. B. Masucci, D. R. Stinson, “Efficient Metering Schemes with Pricing,” IEEE Trans. on IT, Vol.47, pp.2835–2844 (2001)

4. A. De Bonis, B. Masucci, "An Information Theoretic Approach to Metering Scheme," Proc. of ISIT 2000, p.49 (2000)
5. G.J. Simmons, "A Game Theoretical Model of Digital Message Authentication" Congressus Numerantium, Vol.34, pp.413–424 (1982)
6. G.J. Simmons, "Message authentication with arbitration of transmitter/receiver disputes," Proc. of Eurocrypt '87, pp.151–165 (1988)
7. G.J. Simmons, "A Cartesian Product Construction for Unconditionally Secure Authentication Codes that Permit Arbitration," Journal of Cryptology, Vol.2, No.2, pp.77–104 (1990)
8. Y. Wang, R. Safavi-Naini and D. Pei, "Combinatorial Characterisation of  $l$ -Optimal Authentication Codes with Arbitration," J. of Combinatorial Mathematics and Combinatorial Computing, Vol.37, pp.205–224 (2001)
9. T. Johansson, "On the construction of perfect authentication codes that permit arbitration," Proc. of Crypto '93, Lecture Notes in Computer Science, Vol.773, pp.343–354 (1993)
10. T. Johansson, "Lower Bounds on the Probability of Deception in Authentication with Arbitration," IEEE Trans. on Information Theory, Vol.40, pp.1573–1585 (1994)

## A Bounds for Multiple-Use A<sup>2</sup>-code by Wang et al.

Wang, Safavi-Naini and Pei defined the cheating probabilities of  $k$ -multiple-use A<sup>2</sup>-codes as follows, where  $\tilde{P}_t$ ,  $\tilde{P}_{r_i}$  and  $\tilde{P}_{o_i}$  denote the cheating probabilities by  $\mathcal{T}$ ,  $\mathcal{R}$  and  $\mathcal{O}$ , respectively.

$$\begin{aligned}\tilde{P}_t &\triangleq \max_{e_t} \left( \max_{m=(s,a), a \neq e_t(s)} \Pr(e_r(s, a) = 1) \right) \\ \tilde{P}_{r_i} &\triangleq \max_{e_r} \max_{m_1, \dots, m_i} \left( \max_{(s', a') \notin \{m_1, \dots, m_i\}} \Pr(a' = e_t(s') \mid \mathcal{T} \text{ sent } m_1, \dots, m_i) \right) \\ \tilde{P}_{o_i} &\triangleq \max_{m_1, \dots, m_i} \left( \max_{m' \notin \{m_1, \dots, m_i\}} \Pr(\mathcal{R} \text{ accepts } m' \mid \mathcal{T} \text{ sent } m_1, \dots, m_i) \right).\end{aligned}$$

They then showed a lower bound on the cheating probabilities and the size of keys as follows.

**Proposition 2.** [8, Theorem 3.1, 3.2, 3.3]

$$\begin{aligned}\tilde{P}_t &\geq 2^{H(\hat{E}_r | \hat{M}, \hat{E}_t) - H(\hat{E}_r | \hat{E}_t)} \\ \tilde{P}_{r_i} &\geq 2^{-H(\hat{E}_t | \hat{M}_1 \dots \hat{M}_i, \hat{E}_r) + H(\hat{E}_t | \hat{M}_1 \dots \hat{M}_{i+1}, \hat{E}_r)} \\ \tilde{P}_{o_i} &\geq 2^{-H(\hat{E}_r | \hat{M}_1 \dots \hat{M}_i) + H(\hat{E}_r | \hat{M}_1 \dots \hat{M}_{i+1})}\end{aligned}$$

**Proposition 3.** [8, Theorem 4.1, 4.2] If  $\tilde{P}_{o_i}$  and  $\tilde{P}_{r_i}$  achieve their lower bounds, then

$$|E_t| \geq 1 / \prod_{i=0}^{k-1} (\tilde{P}_{o_i} \tilde{P}_{r_i}). \quad (6)$$

If  $\tilde{P}_{o_i}, \tilde{P}_{r_i}, 0 \leq i < k$ , and  $\tilde{P}_t$  achieve their lower bounds, and the equality of eq.(6) holds, then

$$|E_r| \geq 1 / \tilde{P}_t \prod_{i=0}^{k-1} \tilde{P}_{o_i}.$$

## B Construction of Multiple-Use $A^2$ -codes

Wang et al. showed that there exists a  $k$ -multiple-use  $A^2$ -code if there exists a certain combinatorial design [8]. However, they did not show an explicit construction of that design. Therefore, no explicit construction of  $k$ -multiple-use  $A^2$ -code is known.

By substituting the modified Ogata-Kurosawa metering scheme into the proof of Theorem 6, we immediately obtain an explicit construction of a  $k$ -multiple-use  $A^2$ -code as follows.

Let  $p$  be a large prime number.

**Initialization Phase:** An arbiter  $\mathcal{A}'$  chooses a random number  $r \in Z_p$  and two random polynomials  $f_0(y)$  and  $f_1(y)$  with degree at most  $k-1$  over  $GF(p)$ . Let  $e_t = (f_0(y), f_1(y))$  and  $e_r = (r, g(y))$ , where  $g(y) = f_0(y) + rf_1(y)$ . Then  $\mathcal{A}'$  gives  $e_t$  to  $\mathcal{T}$  and  $e_r$  to  $\mathcal{R}$  secretly as their secret keys.

**Communication Phase:** For a source state  $s \in Z_p$ ,  $\mathcal{T}$  sends  $m = (s, f_0(s), f_1(s))$  to  $\mathcal{R}$ .  $\mathcal{R}$  accepts  $m = (s, a, b)$  as authentic iff  $g(s) = a + rb$ .

**Dispute Phase:** On dispute between  $\mathcal{T}$  and  $\mathcal{R}$ ,  $\mathcal{A}'$  accepts  $m = (s, a, b)$  as authentic iff  $f_0(s) = a$  and  $f_1(s) = b$ .

It is clear that  $|E_t| = p^{2k}$ ,  $|E_r| = p^{k+1}$ ,  $|A| = p^2$ . From eq.(3) and Theorem 6, it holds that  $P_T = 1/p$ ,  $P_R \leq 1/p$ ,  $P_O = 1/p$ . More than that, we can show the following lemma.

**Lemma 7.** *In the above  $k$ -multiple-use  $A^2$ -code,  $P_R = 1/p$ .*

*Proof.*  $\mathcal{R}$  has a secret key  $e_r = (r, g(y))$ , where  $g(y) = f_0(y) + rf_1(y)$  for some  $f_0(y)$  and  $f_1(y)$  with degree at most  $k-1$ . Suppose that  $\mathcal{R}$  received  $m_1 = (s_1, a_1), \dots, m_l = (s_l, a_l)$ . Let

$$\begin{aligned} F_0 &\triangleq \{(f_0(y), f_1(y)) \mid g(y) = f_0(y) + rf_1(y), \\ &\quad \text{where } \deg f_0(y) \leq k-1, \deg f_1(y) \leq k-1\}, \\ F_1 &\triangleq \{(f_0(y), f_1(y)) \mid a_1 = (f_0(s_1), f_1(s_1)), \dots, a_l = (f_0(s_l), f_1(s_l))\}. \end{aligned}$$

Then  $\mathcal{R}$  knows that  $e_t \in F_0 \cap F_1$ .

Next suppose that  $\mathcal{R}$  claims that she received  $(s', a')$  such that  $s' \notin \{s_1, \dots, s_l\}$ . If  $m'$  could be made by  $\mathcal{T}$ , then  $a' = (f_0(s'), f_1(s'))$ . Let

$$F_2 \triangleq \{(f_0(y), f_1(y)) \mid a' = (f_0(s'), f_1(s'))\}.$$

Then,

$$\begin{aligned} \Pr(a' = e_t(s')) &= |F_0 \cap F_1 \cap F_2| / |F_0 \cap F_1| \\ &= p^{k-l-1} / p^{k-l} \\ &= 1/p. \end{aligned}$$

□

Then we see that our multiple-use  $A^2$ -code is optimum and Theorem 5 is tight because our multiple-use  $A^2$ -code satisfies all the equalities of Theorem 5.

# Unconditionally Secure Anonymous Encryption and Group Authentication\*

Goichiro Hanaoka<sup>1</sup>, Junji Shikata<sup>2</sup>, Yumiko Hanaoka<sup>3</sup>, and Hideki Imai<sup>1</sup>

<sup>1</sup> Information & Systems, Institute of Industrial Science, University of Tokyo,  
4-6-1 Komaba, Meguro-ku, Tokyo 153-8508, Japan,

hanaoka@iimailab.iis.u-tokyo.ac.jp, imai@iis.u-tokyo.ac.jp

<sup>2</sup> Graduate School of Environment and Information Sciences,

Yokohama National University,

79-7 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan,

shikata@mlab.jks.ynu.ac.jp

<sup>3</sup> Security Systems Laboratory, Multimedia Laboratories, NTT DoCoMo, Inc.,  
3-5 Hikarino-oka, Yokosuka 239-8536, Japan,

claudia@mml.yrp.nttdocomo.co.jp

**Abstract.** Anonymous channels or similar techniques that can achieve sender's anonymity play important roles in many applications. However, they will be meaningless if cryptographic primitives containing his identity is carelessly used during the transmission.

The main contribution of this paper is to study the security primitives for the above problem. In this paper, we first define *unconditionally secure asymmetric encryption scheme* (USAE), which is an encryption scheme with unconditional security and is impossible for a receiver to deduce the identity of a sender from the encrypted message. We also investigate tight lower bounds on required memory sizes from an information theoretic viewpoint and show an optimal construction based on polynomials. We also show a construction based on combinatorial theory, a non-malleable scheme and a multi-receiver scheme. Then, we define and formalize *group authentication code* (GA-code), which is an unconditionally secure authentication code with anonymity like group signatures. In this scheme, any authenticated user will be able to generate and send an authenticated message while the receiver can verify the legitimacy of the message that it has been sent from a legitimate user but at the same time retains his anonymity. For GA-code, we show two concrete constructions.

## 1 Introduction

In many applications, there is a need to allow user or the author of the message to be able to transmit message without revealing his/her identity, e.g. electronic voting. A most commonly used cryptographic technique that is used to build an

---

\* The first author is supported by a Research Fellowship from Japan Society for the Promotion of Science (JSPS).

actual implementation of these characters, is called *anonymous channels* [9,10,1]. However, if not carefully designed, i.e. when a sender uses encryption and authentication methods requiring the sender's identity for decryption or message verification, these systems can be easily compromised, thus corrupting results or violating senders' privacy. For example, if Diffie-Hellman key exchange (with certificates) [13] or (conventional) digital signatures are used, the receiver will be able to easily obtain information regarding the sender's identity, and also may leave the message contents along with the identity of the sender open to perusal. In computationally secure setting, this problem can be solved straightforwardly by using (conventional) public-key encryption e.g. [24,17] and group signatures [11,8] shielding the sender's identity. These schemes and the infrastructure within which they operate are restricted in scope that they rely for their security on the assumed computational difficulty of computing certain number-theoretic problems, such as factoring large composites or solving discrete logarithms in large finite fields. However, this presumption no longer assures the security of computationally secure schemes as the progress in computers as well as further refinement of various algorithms in near future make it computationally able to solve the larger size number-theoretic problems. Unfortunately, in unconditionally secure environment, in which no computational difficulty is assumed, there is yet no straightforward answer to this; all of the current existing schemes use mutual information between sender and receiver, and this mutual information is utilized as a shared communication key between them. This implies that the receiver has to know certain information regarding the sender in prior to selecting a shared secret, and this means, loss of anonymity. (This also implies that *unconditionally secure* public-key encryption scheme is essentially non-existing, since in the model of public-key cryptosystems, a sender and a receiver do not share mutual information between them.) As the increasing computational power approaches where security policy can no longer assume on the difficulty of computationally hard problems, it must shift its focus on assuring the solvency of unconditionally secure schemes that provides long-term security. Similar problem arises in authentication as well. In conventional authentication schemes, the identity of the sender is required for verifying integrity of a transmitted message. In order to protect the sender's privacy in a computationally secure setting, group signatures [11,8] was proposed and since then, group signatures has been greatly studied in the literatures. However, in unconditional setting, there has never existed an authentication scheme that assures anonymity of the sender like that seen in the group signature schemes. For the importance of preparing for the eventual need of long-term security, unconditionally secure setting must be considered a *sine qua non* for a security policy. The main contribution of this paper is to study models, bounds and constructions of novel security primitives on the above problem with no computational assumption. In this paper, we first define *unconditionally secure asymmetric encryption scheme* (USAE), which is an encryption scheme with unconditional security in which a receiver cannot obtain any information of the identity of a sender from the encrypted message. We also investigate tight lower bounds on required memory sizes from information

theory and also show concrete constructions of USAE schemes based on polynomials and cover free family [15]. USAE based on polynomials is optimal due to that it matches the lower bounds. We further show another construction from combinatorial theory, a non-malleable scheme and a multi-receiver scheme. We then, define and formalize *group authentication code*, which is an unconditionally secure authentication code with anonymity like group signatures. In this proposed scheme, any authenticated user will be able to generate an authenticated message and sends it to the receiver. The receiver is then able to verify the authenticity of the received message while maintaining the privacy of the user. Moreover, neither a recipient nor a group authority can obtain any meaningful information of the user who had generated the authenticated message, i.e. no one can link any message to the author who cast it. However, by cooperating with group authority, such as in the case of disputes, the receiver is able to obtain the sender's identity.

### 1.1 Related Works

*Unconditionally Secure Key Distribution Schemes* For confidentiality without computational assumptions, unconditionally secure key distribution schemes are often utilized as suitable security primitives. Blom [5] made the first attempt to construct an unconditionally secure key distribution scheme using MDS linear codes, and his idea was later generalized by Matsumoto and Imai [22], *key predistribution schemes* (KPS), who also proposed a simpler version of KPS, *linear scheme*. Blundo, De Santis, Herzberg, Kutten, Vaccaro and Yung [6] proposed a concrete construction of KPS for conference key distribution and investigated lower bounds on required memory size for users and showed that their scheme, as well as Blom's original scheme and Matsumoto-Imai's scheme, all matched the lower bounds. Blundo, Mattos and Stinson [7], as well as Kurosawa, Yoshida, Desmedt and Burmester [21] showed other interesting bounds on required memory sizes. In [29], in depth survey of various constructions of KPS and corresponding properties has been investigated. KPS may seem to be the best building blocks for unconditionally secure communication systems, however, they are not suitable for certain applications e.g. electronic voting systems, that must ensure user's anonymity; the identity of a sender is required for a recipient to generate the communication key. In all of the existing KPS hitherto, a sender and a receiver's secret information must be used to generate the communication key, and therefore, all of the currently existing schemes does not meet the security requirement for a system with anonymity. As far as we know, there has never existed an unconditionally secure key distribution scheme without a requirement of sender's identity.

*Unconditionally Secure Authentication Schemes and Group Signatures.* For a secure authentication without computational assumptions, unconditionally secure authentication codes (A-codes) [18,27] may be considered which has been intensively studied in the literatures. An overall structure of A-codes is as follows. In the first stage of A-codes, a trusted authority generates secret information

for each of sender and receiver. Then, the sender generates an authenticated message by using his given secret information and transmits it to the receiver. Finally, the receiver verifies the validity of the authenticated message with his secret information. Here, no adversary succeed impersonation nor substitution attack even if the adversary has unlimited computational power. There has also been many attempts to modify A-codes with the aim of enhancing the codes with desirable properties other than anonymity, such as asymmetry [28] and multireceiver-authenticity [12]. However, in none of these attempted modifications, receivers were able to identify the sender of the message. Thus, there were no existing A-codes and their variants that were applicable concerning the protection of the sender's identity, i.e. no anonymity. Though there are some unconditionally secure digital signature schemes [19,20,26] that do exist, these schemes yet too, do not provide anonymity. However, in computationally secure settings, anonymity can be achieved by using *group signatures* [11,8]. For a group signature, a user is able to prove that he is a legitimate user of the group by using his secret information given by a group authority, and in the case of a dispute, the group authority can identify the user from a published signature signed by the user. Group signature is therefore a suitable authenticating scheme that can be used especially in case where the privacy of the user has to be maintained. However, all the existing group signature schemes are based on computational assumptions and will be broken if certain computationally hard problems, e.g. discrete logarithm or factoring, are solved.

## 1.2 Our Results

We start this paper by defining *unconditionally secure asymmetric encryption scheme* (USAE) with formal definitions. USAE is an encryption scheme with unconditional security in which a receiver cannot gain any information of a particular user from an encrypted message. We investigate from information theory, the lower bounds for the required memory sizes of a ciphertext, a sender and a receiver's secrets. Further, we propose concrete constructions of USAE based on polynomials and also constructions based on cover free families. Polynomial based construction is optimal due to that it matches the lower bounds which in turn implies that the lower bounds are all tight. One important fact to mention, it is remarkable that these bounds that we show are considerably different from those in Shannon's model for conventional unconditionally secure symmetric encryption. Comparison between polynomial-based and cover free family-based schemes are also made. In addition, we study an extension of USAE, that with non-malleability. More precisely, a formal definition of non-malleability, a concrete non-malleable scheme and a security proof are investigated. Furthermore, another extension of USAE, for multiple-receiver setting, is shown. We continue by defining *group authentication code* (GA-code) with formal definitions. GA-code is an unconditionally secure authentication code with anonymity like group signatures. In GA-codes, any user in a group can generate an authenticated message and verify it as long as it has been sent from a legitimate user in a group. Moreover, a receiver is not able to obtain any meaningful information of a partic-



ular user who had generated the authenticated message. However, in the case of disputes, a receiver is able to obtain the sender's identity by cooperating with a group authority. It is important to note here that group authority or the receiver by itself will be insufficient in obtaining to obtain any information regarding the user i.e. they must cooperate. We then show two concrete constructions of GA-code with formal security proofs. One construction is based on polynomials and the other on cover free families and A-codes. Organization of this paper is as follows: In section 2, we study the model, bounds and constructions for USAE. Polynomial based USAE construction is optimal due to that it matches the lower bounds. This in turn implies that the lower bounds are all tight. We also show other efficient and secure implementations of USAE. In section 3, we show model and concrete construction for GA-code with formal security proof.

## 2 Unconditionally Secure Asymmetric Encryption

In this section, model, security definition, lower bounds and concrete constructions of USAE are shown. One of our constructions is optimal in terms of required memory sizes for a ciphertext, an encryption key and a decryption key. It should be noted that the definition that we use for “asymmetric encryption” in this paper is not equivalent to the meaning of “public-key encryption” in a general sense. Here, in USAE, “asymmetric” is used as a pair of encryption and decryption keys that are asymmetric, where an encryption key is not public.

### 2.1 Model

Since no computational difficulty is assumed in USAE, it is impossible for a sender to secretly transmit a message using only the public information. This means that in order to construct a USAE, a different assumption (rather than computational assumptions) will be required, e.g. existence of a noisy channel, that of a quantum channel, bounds of memory or threshold of the number of malicious users. For simplicity, we introduce the *trusted initializer model* [23], in which we assume a trusted initializer who honestly distributes each user's secret in the initial phase and deletes his memory after the distribution of the secrets. We should note that the trusted initializer can be removed by using multi-party computation [4] if the number of malicious users is less than a third of the total number of users and there exists a private channel between each pair of users.

In the model of USAE, there are  $n + 2$  participants, a set of  $n$  senders  $\{S_1, \dots, S_n\}$ , a receiver  $R$  and a trusted initializer TI. TI generates encryption keys  $e_1, \dots, e_n$  for  $S_1, \dots, S_n$ , respectively, and a decryption key  $d$  for  $R$ . After distributing these keys, TI deletes his memory. In order to send a plaintext  $m$  to  $R$  with confidentiality,  $s \in \{S_1, \dots, S_n\}$  encrypts  $m$  by using  $e_i$  and transmits a ciphertext  $c$  to  $R$ .  $R$  decrypts  $c$  by using  $d$  and recovers  $m$ .

## 2.2 Definition

Here, we formally define the security of USAE. It should be noted that, in addition to confidentiality, anonymity of a sender is required for USAE. Let  $\mathcal{S}$ ,  $\mathcal{E}_i$  ( $i = 1, \dots, n$ ),  $\mathcal{D}$ ,  $\mathcal{M}$  and  $\mathcal{C}$  denote the random variables induced by  $s$ ,  $e_i$  ( $i = 1, \dots, n$ ),  $d$ ,  $m$  and  $c$ , respectively. For a random variable  $\mathcal{X}$ ,  $H(\mathcal{X})$  denotes the entropy of  $\mathcal{X}$ . For  $\mathcal{X}$ , let  $X := \{x | \Pr(\mathcal{X} = x) > 0\}$ .  $|X|$  denotes the cardinality of  $X$ . We assume that at most  $k$  ( $0 \leq k \leq n - 1$ ) authorized senders are malicious. Then, the security of USAE is formally defined as follows:

**Definition 1.** We say that  $(\mathcal{E}_1, \dots, \mathcal{E}_n, \mathcal{D}, \mathcal{M}, \mathcal{C})$  is a  $(k, n)$ -one-time USAE if

1.  $R$  can correctly decrypt  $m$  from  $c$ , that is,  $H(\mathcal{M}|\mathcal{C}, \mathcal{D}) = 0$ .
2. Any set of  $k$  malicious senders has no information on  $m$  from  $c$ . Namely, for any set of  $k$  malicious senders  $\{S_{i_1}, \dots, S_{i_k}\} \subset \{S_1, \dots, S_n\}$  such that  $s \notin \{S_{i_1}, \dots, S_{i_k}\}$ ,  $H(\mathcal{M}|\mathcal{C}, \mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k}) = H(\mathcal{M})$ .
3.  $R$  obtains no information on the identity of  $s$  from  $c$ . Namely,  $H(\mathcal{S}|\mathcal{C}) = H(\mathcal{S})$ .
4. Additionally, we assume that a ciphertext  $c$  is uniquely determined from a plaintext  $m$  and an encryption key  $e_i$ , i.e.  $H(\mathcal{C}|\mathcal{M}, \mathcal{E}_i) = 0$  for any  $i$ .

## 2.3 Lower Bounds

In this subsection, lower bounds on required memory sizes for a ciphertext, an encryption key and a decryption key in USAE are shown. These bounds are all tight since we also show a construction which matches them (see section 2.4, for details). Note that proofs of Theorem 1, 3 and Lemma 1 are omitted, and will appear in the full version of this paper. We begin by showing a lower bound on the required memory size for a ciphertext.

**Theorem 1.** In a  $(k, n)$ -one-time USAE,  $H(\mathcal{C}) \geq H(\mathcal{M}) + H(\mathcal{S})$ .

Theorem 1 implies that the required memory size for a ciphertext is always larger than that for a plaintext by at least  $H(\mathcal{S})$  bits. Next is a lemma that shows the relationship between the required memory size for an encryption key  $e_i$  and for a ciphertext  $c$  in USAE.

**Lemma 1.** In a  $(k, n)$ -one-time USAE,  $H(\mathcal{E}_i) \geq H(\mathcal{C})$ , for any  $i$ .

Lemma 1 implies that the memory size requirement for an encryption key in USAE is equal or greater than that for a ciphertext. This is also closely related to the famous Shannon's result [25]. That is, in unconditionally secure symmetric encryption, it is a well-known fact that the required memory size for an encryption key is equal or greater than that for a plaintext, assuming that a ciphertext is uniquely determined from a plaintext and an encryption key. Now, a lower bound on the required memory size for an encryption key is shown.

**Theorem 2.** In a  $(k, n)$ -one-time USAE,  $H(\mathcal{E}_i) \geq H(\mathcal{M}) + H(\mathcal{S})$ , for any  $i$ .

*Proof.* From Lemma 1 and Theorem 1, we have  $H(\mathcal{E}_i) \geq H(\mathcal{M}) + H(\mathcal{S})$  for any  $i$ .  $\square$

Theorem 2 implies that the required memory size for an encryption key is always larger than that for a plaintext by at least  $H(\mathcal{S})$  bits. Finally, we show a lower bound on the required memory size for a decryption key.

**Theorem 3.** *In a  $(k, n)$ -one-time USAE,  $H(\mathcal{D}) \geq (k + 1)H(\mathcal{M})$  if the equality in Lemma 1 is satisfied for any  $i$ .*

Theorem 3 implies that the required memory size for a decryption key is  $(k + 1)$  times larger than that for a plaintext.

## 2.4 Constructions

Now, we show two concrete constructions of USAE. One of the constructions is based on polynomials over finite fields, and the other one on cover free family [15]. The polynomial based construction is optimal in terms of required memory sizes for a ciphertext, an encryption key and a decryption key. For the cover free family construction, security parameters can be flexibly determined.

In this subsection, we assume that the distribution of the sender is uniform, that is,  $\Pr(\mathcal{S} = S_i) = \frac{1}{n}$  for any  $i$  ( $1 \leq i \leq n$ ).

**OPTIMAL CONSTRUCTION FROM POLYNOMIALS.** Here, we show an optimal  $(k, n)$ -one-time USAE which meets all our bounds. This means that the lower bounds in the previous subsection are all tight.

**Definition 2.** A  $(k, n)$ -one-time USAE is *optimal* if one has equalities in Theorem 1, 2 and 3.

### Optimal $(k, n)$ -One-Time USAE Based on Polynomials

**1. Setting Up:** Let  $|M| = q$ , where  $q$  is a prime power and  $q \geq n$ . TI chooses a uniformly random polynomial  $f(x) = \sum_{i=0}^k a_i x^i$  over  $GF(q)$ . TI also chooses distinct numbers  $b_i$  ( $1 \leq i \leq n$ ) from a set  $B \subseteq GF(q)$  uniformly at random, where  $|B| = n$ .  $B$  may be public to all players. Next, TI gives  $f(x)$  to  $R$  as his decryption key, and also gives  $\{b_1, f(b_1)\}, \{b_2, f(b_2)\}, \dots, \{b_n, f(b_n)\}$  to  $S_1, S_2, \dots, S_n$  as encryption keys, respectively. TI deletes his memory after distributing the keys.

**2. Encryption:** Sender  $S_i$  encrypts  $m$  by  $c = \{b_i, c'\}$ , where  $c' := f(b_i) + m$ .

**3. Decryption:** Receiver  $R$  decrypts  $c$  by  $f(x)$  as follows:  $m = c' - f(x)|_{x=b_i}$ .

**Theorem 4.** *The above scheme is an optimal  $(k, n)$ -one-time USAE.*

*Proof.* In the above scheme,  $H(\mathcal{C}) = H(\mathcal{M}) + \log_2 n$ ,  $H(\mathcal{E}_i) = H(\mathcal{M}) + \log_2 n$  ( $1 \leq i \leq n$ ) and  $H(\mathcal{D}) = (k + 1)H(\mathcal{M})$ . It is clear that the above scheme satisfies the first condition of Def. 1. Suppose that colluders  $S_{i_1}, \dots, S_{i_k}$ , such that  $S_i \notin \{S_{i_1}, \dots, S_{i_k}\}$ , can obtain certain information on  $m$  from  $c$ . This implies that the colluders has certain information on  $f(b_i)$ . However, this is impossible because  $\deg f(x) = k$  and the colluders knows only the  $k$  points of  $f(x)$ . Hence, the above

scheme satisfies the second condition of Def. [1](#). Finally, since, for a ciphertext  $c = \{b, c'\}$  such that  $b \in B$  and  $c' = f(b) + m$ , any of  $S_1, \dots, S_n$  can be a possible sender of the ciphertext from  $R$ 's point of view, and therefore,  $R$  can determine who the sender of the ciphertext is with probability at most  $1/n$ . Hence, the above scheme satisfies the third condition of Def. [1](#) as well.  $\square$

**CONSTRUCTION FROM COVER FREE FAMILY.** Here, we show a construction of USAE based on cover free family [\[15\]](#) which allows a more flexible parameter setting than the polynomial based one. Namely, in cover free family based construction, it is possible to choose parameters  $n$  and  $|M|$  with  $|M| < n$ , while, in polynomial based construction, these two parameters must always be determined to be  $|M| \geq n$ .

**Definition 3.** Let  $L := \{\ell_1, \ell_2, \dots, \ell_t\}$  and  $F = \{F_1, \dots, F_n\}$  be a family of subsets of  $L$ . We call  $(L, F)$  an  $(n, t, k)$  *cover free family* (CFF) if  $F_0 \not\subseteq F_1 \cup \dots \cup F_k$  for all  $F_0, F_1, \dots, F_k \in F$ , where  $F_i \neq F_j$  if  $i \neq j$ .

A trivial CFF is the family consisting of single element subsets, in which case  $n = t$ . It should also be noted that there exist nontrivial constructions of CFF with  $n > t$ . Construction of CFFs is intensively studied in various areas of mathematics such as finite geometry, design theory, and probability theory. Concrete methods for generating CFF are given in [\[16\]](#).

#### **$(k, n)$ -One-Time USAE Based on $(n, t, k)$ -CFF**

**1. Setting Up:** TI first generates an  $(n, t, k)$ -CFF such that each of  $\ell_i$  ( $1 \leq i \leq t$ ) is an element of  $GF(q)$ , where  $\mathcal{M} = GF(q)$ . TI also chooses distinct numbers  $r_i$  ( $1 \leq i \leq n$ ) from  $\{1, 2, \dots, n\}$  uniformly at random. An algorithm that generates  $F_i$  ( $1 \leq i \leq n$ ) from  $i$  and  $L$  may be public to all players. Next, TI gives  $L$  to  $R$  as his decryption key. TI also gives  $\{r_i, \ell^{(i)}\}$  ( $1 \leq i \leq n$ ) to  $S_i$  ( $1 \leq i \leq n$ ), respectively, as encryption keys, where  $\ell^{(i)} := \sum_{\ell \in F_{r_i}} \ell$ . After distributing the keys, TI deletes his memory.

**2. Encryption:** Sender  $S_i$  encrypts  $m$  by  $c = \{r_i, c'\}$ , where  $c' = m + \ell^{(i)}$ .

**3. Decryption:** Receiver  $R$  generates  $F_{r_i}$  from  $L$  and  $r_i$ . Then,  $R$  computes  $m$  as  $m = c' - \sum_{\ell \in F_{r_i}} \ell$ .

**Theorem 5.** *The above scheme is a  $(k, n)$ -one-time USAE.*

*Proof.* It is obvious that the above scheme satisfies all of conditions in Def. [1](#).  $\square$

The required memory sizes for the above construction is formally addressed as follows:

**Theorem 6.** *The required memory sizes in the above construction are given as follows:*

$$H(\mathcal{C}) = \log_2 nq, \quad H(\mathcal{E}_i) = \log_2 nq \quad \text{for any } i \ (1 \leq i \leq n), \quad H(\mathcal{D}) = t \log_2 q.$$

It should be noted that the cover free family based construction matches the lower bounds on the required memory sizes for a ciphertext and an encryption key.

COMPARISON. Here, comparison between polynomial and cover free family based constructions is explored. Given the fact described above, our polynomial construction is optimal in terms of required memory sizes. Therefore, polynomial based construction is theoretically superior to the cover free family based construction storage wise. However, polynomial based construction can only be implemented when  $|M| \geq n$  although, in most practical situations, this restriction may be ignored. On the other hand, for the cover free family based construction, it allows even for  $|M| < n$  when there exist an appropriate cover free family. We now show an example of system parameter settings in the case when this restriction do, applies. For the following situation, the cover free family based construction will be more suitable than polynomial based construction in terms of required memory sizes.

*Example.* Assume that the message space is  $\{yes, no\}$  and we need a (127, 128)-one-time USAE. For the polynomial based construction, a finite field  $GF(q)$  with  $q \geq 128$  is required. Consequently, the size of a ciphertext will be at least 14 bits. A receiver and a sender must then store at least 896 bits and 14 bits, respectively. For the cover free family based construction, (128, 128, 127)-CFF (trivial CFF) over  $GF(2)$ , the size of a ciphertext will be 8 bits at the least, and a receiver and a sender store at least 128 bits and 8 bits, respectively. For the described situation, we can see a significant advantage of the cover free family based construction over the polynomial based construction.

In summary, different constructions are advantageous for different perspectives, so, one construction may do better than another under certain circumstances. However, the polynomial based construction is generally most suitable for typical security parameter settings in USAE. And for the case when  $|M| < n$ , the cover free family based construction betters.

Memory sizes requirement can be reduced further for the above example if we utilize nontrivial CFF instead. However, in a nontrivial CFF, the number of malicious senders cannot be set to a considerably larger number than the total number of the senders. This fact is due to the following proposition:

**Proposition 1** ([16]). *In a nontrivial  $(n, t, k)$ -CFF with  $n > t$ ,  $\frac{k(k-1)}{2} \leq n$ .*

## 2.5 Extensions

NON-MALLEABLE SCHEME. Here, we consider *non-malleability* [14] of the proposed USAE. Frankly, non-malleability means an adversary's inability: given a challenge ciphertext  $c$ , to generate a different ciphertext  $\hat{c}$  such that the plaintexts  $m, \hat{m}$  underlying these two ciphertexts are meaningfully related. For computational encryption schemes, formal definitions of non-malleability are given in [23]. Here, we give a definition of non-malleability for USAE.

**Definition 4.** Let  $\hat{c}(\neq c)$  be another ciphertext which could have been generated by  $s$  instead of  $c$  in USAE, and  $\hat{m}(\neq m)$  be a plaintext underlying  $\hat{c}$ . Let  $\hat{C}$

and  $\hat{\mathcal{M}}$  denote random variables induced by  $\hat{c}$  and  $\hat{m}$ , respectively. A USAE is *perfectly non-malleable* if the following equation holds:

$$H(\hat{\mathcal{M}}|\mathcal{C}, \hat{\mathcal{C}}, \mathcal{M}, \mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k}) = H(\hat{\mathcal{M}}|\mathcal{C}, \mathcal{M}), \quad (1)$$

for any set of  $k$  malicious senders  $\{S_{i_1}, \dots, S_{i_k}\} \subset \{S_1, \dots, S_n\}$  such that  $s \notin \{S_{i_1}, \dots, S_{i_k}\}$ .

The above definition is reasonable since Eq. [1](#) implies that even if an adversary knows a pair of  $\{c, m\}$ , there is no other ciphertext which can give further information except the information that its underlying plaintext is not identical to  $m$ . In other words, no adversary can generate a ciphertext whose plaintext is meaningfully related to  $m$  when Eq. [1](#) holds.

A USAE which satisfies perfect non-malleability is constructed as follows:

### Non-malleable $(k, n)$ -One-Time USAE Based on Polynomials

**1. Setting Up:** Let  $|M| = q$ , where  $q$  is a prime power and  $q \geq n$ . TI chooses a uniformly random polynomials  $f_i(x) = \sum_{j=0}^k a_{ij}x^j$  ( $i = 1, 2$ ) over  $GF(q)$ . TI also chooses distinct numbers  $b_i$  ( $1 \leq i \leq n$ ) from a set  $B \subseteq GF(q)$  uniformly at random, where  $|B| = n$ , such that  $f_2(b_i) \neq 0$  for any  $i$  ( $1 \leq i \leq n$ ).  $B$  may be public to all players. Next, TI gives  $f_1(x)$  and  $f_2(x)$  to  $R$  as his decryption key, and also gives  $\{b_1, f_1(b_1), f_2(b_1)\}, \{b_2, f_1(b_2), f_2(b_2)\}, \dots, \{b_n, f_1(b_n), f_2(b_n)\}$  to  $S_1, S_2, \dots, S_n$  as encryption keys, respectively. TI deletes his memory after distributing the keys.

**2. Encryption:** Sender  $S_i$  encrypts  $m$  by  $c = \{b_i, c'\}$ , where  $c' := f_1(b_i) + mf_2(b_i)$ .

**3. Decryption:** Receiver  $R$  decrypts  $c$  by  $f_1(x)$  and  $f_2(x)$  as follows:  $m = (c' - f_1(x)|_{x=b_i})/f_2(x)|_{x=b_i}$ .

**Theorem 7.** *The above scheme is a perfectly non-malleable  $(k, n)$ -one-time USAE.*

*Proof.* Similarly to the proof of Theorem [4](#) it can be proved that the above scheme is a  $(k, n)$ -one-time USAE. Now, we show that the above scheme satisfies the equality of Eq. [1](#). It is obvious that

$$\begin{aligned} H(\hat{\mathcal{M}}|\mathcal{C}, \mathcal{M}) = & - \sum_{m \in M} \sum_{\hat{m} \in M \setminus \{m\}} \Pr(\mathcal{M} = m) \Pr(\hat{\mathcal{M}} = \hat{m}|\mathcal{M} = m) \\ & \cdot \log_2 \Pr(\mathcal{M} = m) \Pr(\hat{\mathcal{M}} = \hat{m}|\mathcal{M} = m). \end{aligned} \quad (2)$$

Next, we show that  $H(\hat{\mathcal{M}}|\mathcal{C}, \hat{\mathcal{C}}, \mathcal{M}, \mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k})$  is equivalent to that in Eq. [2](#). Since both  $\deg f_1(x)$  and  $\deg f_2(x)$  are  $k$ , no information on  $f_1(x)$  and  $f_2(x)$  cannot be obtained even if  $e_1, \dots, e_k$  are used. Then, a set of all possible values for  $(f_1(x), f_2(x))$  becomes  $\Gamma := \{(\gamma_1, \gamma_2) | c' = \gamma_1 + m\gamma_2, \gamma_2 \neq 0\}$ . Consequently, for given  $\hat{c} (= \{b_i, \hat{c}'\})$ , a set of all possible plaintext  $\hat{m}$  underlying  $\hat{c}$  is  $M' := \{m' | m' = (\hat{c}' - \gamma_1)/\gamma_2, \forall (\gamma_1, \gamma_2) \in \Gamma\}$ . From Lemma [2](#) and [3](#) we have  $M' = M \setminus \{m\}$  and a mapping  $\tau : \Gamma \rightarrow M'$ , such that  $\tau(\gamma_1, \gamma_2) = (\hat{c}' - \gamma_1)/\gamma_2$ , is

bijective. Hence, we have

$$\begin{aligned}
& H(\hat{\mathcal{M}}|\mathcal{C}, \hat{\mathcal{C}}, \mathcal{M}, \mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k}) \\
&= - \sum_{m \in M} \sum_{(\gamma_1, \gamma_2) \in \Gamma} \Pr(\mathcal{M} = m) \Pr(\hat{\mathcal{M}} = \tau(\gamma_1, \gamma_2) | \mathcal{M} = m) \\
&\quad \cdot \log_2 \Pr(\mathcal{M} = m) \Pr(\hat{\mathcal{M}} = \tau(\gamma_1, \gamma_2) | \mathcal{M} = m) \\
&= - \sum_{m \in M} \sum_{\hat{m} \in M \setminus \{m\}} \Pr(\mathcal{M} = m) \Pr(\hat{\mathcal{M}} = \hat{m} | \mathcal{M} = m) \\
&\quad \cdot \log_2 \Pr(\mathcal{M} = m) \Pr(\hat{\mathcal{M}} = \hat{m} | \mathcal{M} = m). \quad (3)
\end{aligned}$$

From Eq. 2 and 3, Eq. 1 holds.  $\square$

**Lemma 2.** *For a given ciphertext  $c(= \{b_i, c'\}) \in C$  and its corresponding plaintext  $m \in M$ , let  $\Gamma := \{(\gamma_1, \gamma_2) | c = \gamma_1 + m\gamma_2, \gamma_2 \neq 0\}$ . Then, for any  $\hat{c}(= \{b_i, \hat{c}'\}) \in C$ , such that  $\hat{c}' \neq c'$ ,  $(\hat{c}' - \gamma_1)/\gamma_2 \neq m$  if  $(\gamma_1, \gamma_2) \in \Gamma$ .*

*Proof.* Suppose that there exist  $(\gamma_1, \gamma_2) \in \Gamma$ , such that  $(\hat{c}' - \gamma_1)/\gamma_2 = m$ . Then,  $\hat{c}' = \gamma_1 + m\gamma_2 = c'$ . Since  $\hat{c}' \neq c'$ , this is a contradiction.  $\square$

**Lemma 3.** *For a given ciphertext  $c(= \{b_i, c'\}) \in C$  and its corresponding plaintext  $m \in M$ , let  $\Gamma := \{(\gamma_1, \gamma_2) | c = \gamma_1 + m\gamma_2, \gamma_2 \neq 0\}$ . Then, for any  $\hat{c}(= \{b_i, \hat{c}'\}) \in C$ , such that  $\hat{c}' \neq c'$ ,  $(\hat{c}' - \gamma_{11})/\gamma_{12} \neq (\hat{c}' - \gamma_{21})/\gamma_{22}$  if  $(\gamma_{11}, \gamma_{12}) \neq (\gamma_{21}, \gamma_{22})$  and  $(\gamma_{11}, \gamma_{12}), (\gamma_{21}, \gamma_{22}) \in \Gamma$ .*

*Proof.* Suppose that there exist  $(\gamma_{11}, \gamma_{12}), (\gamma_{21}, \gamma_{22}) \in \Gamma$ , such that  $(\hat{c}' - \gamma_{11})/\gamma_{12} = (\hat{c}' - \gamma_{21})/\gamma_{22}$ . Letting  $\hat{m} := (\hat{c}' - \gamma_{11})/\gamma_{12} (= (\hat{c}' - \gamma_{21})/\gamma_{22})$ , we have  $\hat{c}' = \gamma_{11} + \hat{m}\gamma_{12} = \gamma_{21} + \hat{m}\gamma_{22}$ . Hence,

$$(\gamma_{11} - \gamma_{21}) = -\hat{m}(\gamma_{12} - \gamma_{22}). \quad (4)$$

Also, since  $c' = \gamma_{11} + m\gamma_{12} = \gamma_{21} + m\gamma_{22}$ , it is clear that

$$(\gamma_{11} - \gamma_{21}) = -m(\gamma_{12} - \gamma_{22}). \quad (5)$$

From Eq. 4 and 5, it is obvious that  $(\gamma_{11} - \gamma_{21}) = (\gamma_{12} - \gamma_{22}) = 0$  or  $m' = m$ . When  $(\gamma_{11} - \gamma_{21}) = (\gamma_{12} - \gamma_{22}) = 0$ , we get  $(\gamma_{11}, \gamma_{12}) = (\gamma_{21}, \gamma_{22})$ . This is a contradiction. On the other hand, when  $m' = m$ , this is also a contradiction due to Lemma 2.  $\square$

**MULTIPLE-RECEIVER SCHEME.** The model of USAE described in section 2.1 is built for a single receiver. That is, there exists only one receiver for the entire model. From this, we can extend the model to be a multiple receiver model and show an efficient implementation of it. More detailed discussion will be provided in the full version of this paper.

### 3 Group Authentication Code

In this section, we show a model, security definition and a concrete construction of GA-code, which is an unconditionally secure authentication code with anonymity like group signatures. With the combination of USAE and GA-code, a secure communication system, which assures confidentiality, authenticity and user's anonymity can be constructed without any computational assumptions.

#### 3.1 Model

Similar to what we saw in the model of USAE, we introduce the *trusted initializer model* for GA-code as well. In GA-code model, there are  $n + 3$  participants, a set of  $n$  senders  $\{S_1, \dots, S_n\}$ , a receiver  $R$ , a group authority  $G$  and a trusted initializer, TI. TI generates secret information  $u_1, \dots, u_n$  for  $S_1, \dots, S_n$ , respectively, and secret information  $v$  for  $R$ . TI also generates secret information  $w$  for  $G$ . After distributing these keys, TI deletes his memory. In order to send a plaintext  $m$  to  $R$  with authenticity,  $s \in \{S_1, \dots, S_n\}$  generates an authenticated message  $\alpha$  from  $m$  by using  $u_i$  and transmits  $\alpha$  to  $R$ .  $R$  verifies the validity of  $\alpha$  by using  $m$  and  $v$ . In a situation where  $R$  wants to reveal the identity of the sender,  $R$  can obtain it by cooperating with  $G$  only if  $G$  approves  $R$ 's request.

#### 3.2 Definition

Here, we formally define the security of GA-code. In GA-code, a sender is able to prove that he is a legitimate member of a group,  $\{S_1, \dots, S_n\}$ . In addition, by cooperating with  $G$ ,  $R$  can obtain the identity of the sender fairly simply. However, each of  $R$  and  $G$  alone, cannot reveal the sender's identity.

An adversary can perform *impersonation* or *substitution* by constructing a fraudulent codeword. The attack is considered successful if the receiver accepts the codeword. In impersonation, an adversary is assumed to not have seen any communication occurred priorly, while in substitution, the adversary have seen at least one transmitted codeword. Both impersonation and substitution can be performed by either senders and outsiders, where none of TI,  $G$ ,  $R$ ,  $S_1, \dots, S_n$  is included in the collusion of the outsiders. Also, senders' attack is considered to be successful if a fraudulent codeword is accepted by the receiver and no fraudulent message is traced back to the malicious sender who wrote the message by the receiver and a group authority. Outsiders' attack is considered successful if the receiver accepts the fraudulent codeword. Note that, mixed collusion attack delivered together by senders and outsiders is referred to an attack made only by senders.

Let  $\mathcal{S}, \mathcal{U}_i$  ( $i = 1, \dots, n$ ),  $\mathcal{V}, \mathcal{W}, \mathcal{M}$  and  $\mathcal{A}$  denote the random variables induced by  $s, u_i$  ( $i = 1, \dots, n$ ),  $v, w, m$  and  $\alpha$ , respectively. For  $\mathcal{X}$ , let  $X := \{x | \Pr(\mathcal{X} = x) > 0\}$ .  $|X|$  denotes the cardinality of  $X$ .

We assume that at most  $k$  ( $0 \leq k \leq n - 1$ ) authorized senders are malicious. Then, the security of GA-code is formally defined as follows:



**Definition 5.** We say that  $(\mathcal{U}_1, \dots, \mathcal{U}_n, \mathcal{V}, \mathcal{W}, \mathcal{M}, \mathcal{A})$  is a  $(p, k, n)$ -one-time group authentication code (GA-code) if

1. Any set of  $k$  malicious senders can perform impersonation with probability at most  $p$ . Namely, for any set of  $k$  malicious senders  $\{S_{i_1}, \dots, S_{i_k}\} \subset S$ ,

$$\max_{u_{i_1}, \dots, u_{i_k}} \max_{\alpha} \Pr(R \text{ accepts } \alpha \wedge \text{none of } \{S_{i_1}, \dots, S_{i_k}\} \text{ is detected as the sender of } \alpha | u_{i_1}, \dots, u_{i_k}) \leq p.$$

2. Any outsiders can perform impersonation with probability at most  $p$ , i.e.  $\max_{\alpha} \Pr(R \text{ accepts } \alpha) \leq p$ .
3. Any set of  $k$  malicious senders can perform substitution with probability at most  $p$ . Namely, letting  $\mathcal{S} = S_{i_0}$ , for any set of  $k$  malicious senders  $S_{i_1}, \dots, S_{i_k}$  such that  $S_{i_0} \notin \{S_{i_1}, \dots, S_{i_k}\}$ ,

$$\max_{u_{i_1}, \dots, u_{i_k}} \max_{\alpha'} \max_{\alpha, \alpha \neq \alpha'} \Pr(R \text{ accepts } \alpha \wedge \text{none of } \{S_{i_1}, \dots, S_{i_k}\} \text{ is detected as the sender of } \alpha | u_{i_1}, \dots, u_{i_k}, \alpha') \leq p,$$

where  $\alpha'$  is taken over the set of valid authenticated messages which can be generated by  $S_{i_0}$ .

4. Any set of outsiders can perform substitution with probability at most  $p$ , i.e. letting  $\alpha'$  be an authenticated message which is generated by an honest user,  $\max_{\alpha'} \max_{\alpha, \alpha \neq \alpha'} \Pr(R \text{ accepts } \alpha | \alpha') \leq p$ .
5.  $R$  obtains no information on the identity of  $s$  from  $\alpha$ , namely,  $\Pr(\mathcal{S} = S_i | \alpha, v) = \Pr(\mathcal{S} = S_i)$  for any  $\alpha$  and  $i$  ( $1 \leq i \leq n$ ).
6.  $G$  obtains no information on the identity of  $s$  from  $\alpha$ , namely,  $\Pr(\mathcal{S} = S_i | \alpha, w) = \Pr(\mathcal{S} = S_i)$  for any  $\alpha$  and  $i$  ( $1 \leq i \leq n$ ).
7. Cooperating with  $G$ ,  $R$  can reveal the identity of the sender of the authenticated message  $\alpha$  with probability more than  $\Pr(\mathcal{S} = S_{i_0})$ , where  $S_{i_0}$  is the sender of  $\alpha$ .

### 3.3 Constructions

In this subsection, we show a couple of constructions of GA-codes; one is based on polynomials and the other is based on cover free families.

**CONSTRUCTION FROM POLYNOMIALS.** Based on polynomials, a GA-code can be constructed as follows:

#### GA-Code Based on Polynomials

**1. Setting Up:** Let  $M = GF(q) \setminus \{0\}$ , where  $q$  is a prime power and  $q \geq n$ . TI chooses a uniformly random polynomials  $f(x)$  and  $g(x)$  over  $GF(q)$  such that  $\deg f(x) \leq k+1$  and  $\deg g(x) \leq k+1$ . TI also chooses distinct numbers  $b_i$  ( $1 \leq i \leq n$ ) from  $B \subseteq GF(q)$  uniformly at random, where  $|B| = n$  such that  $f(b_i) \neq f(b_j)$  for any  $i, j$  with  $1 \leq i, j \leq n$ ,  $i \neq j$ . Next, TI gives  $f(x)$  and  $g(x)$  to

$R$  as  $v$ , and also gives  $\{b_1, f(b_1), g(b_1)\}, \{b_2, f(b_2), g(b_2)\}, \dots, \{b_n, f(b_n), g(b_n)\}$  to  $S_1, S_2, \dots, S_n$  as  $u_1, u_2, \dots, u_n$ , respectively. TI also generates a mapping  $\pi : GF(q) \rightarrow S$  such that  $\pi(f(b_i)) = S_i$  and gives it to  $G$  as  $w$ . TI deletes his memory after distributing the keys.

**2. Message Generation:** Sender  $S_i$  generates an authenticated message  $\alpha$  for  $m$  as  $\alpha = \{m, b_i, h\}$ , where  $h := f(b_i)m + g(b_i)$ .

**3. Verification:** Receiver  $R$  accepts  $\alpha$  as valid if  $h$  is identical to  $f(x)|_{x=b_i}m + g(x)|_{x=b_i}$ .

**4. Tracing:** When  $R$  wants to reveal the identity of the sender,  $R$  first sends a request to  $G$ . If  $R$ 's request is approved by  $G$ ,  $R$  transmits  $f(b_i)$  to  $G$  via a secure channel. Then,  $G$  reveals the sender's identity by  $S_i = \pi(t)$  and transmits this result back to  $R$ .

The security of the above scheme is addressed as follows:

**Lemma 4.** *In the GA-code based on polynomials, colluders, which include at most  $k$  of  $\{S_1, S_2, \dots, S_n\}$ , can perform impersonation with probability at most  $\frac{1}{q}$ . (See conditions 1 and 2 of Def. 5.)*

*Proof.* For succeeding impersonation by collusion of senders  $\{S_{i_1}, \dots, S_{i_k}\}$ , adversaries need to produce a fraudulent message  $\{b, m', h'\}$  such that  $h' = f(b)m' + g(b)$  and  $b \notin \{b_{i_1}, \dots, b_{i_k}\}$ . Since the malicious senders have only  $k$  points of  $g(x)$ , it is therefore, impossible to obtain any information on  $g(b)$ , and accordingly, they also have no information on  $h'$ . Therefore, the probability of succeeding the attack will be at most  $1/q$ . In similar manner to this, we can also prove that the probability of succeeding outsiders' impersonation is at most  $1/q$ .  $\square$

**Lemma 5.** *In the GA-code based on polynomials, colluders, which include at most  $k$  of  $\{S_1, S_2, \dots, S_n\} \setminus \{S_{i_0}\}$ , can perform substitution with probability at most  $\frac{1}{q-k}$ , where  $S_{i_0}$  is the honest sender who sends a valid authenticated message  $\alpha'$  to  $R$ . (See conditions 3 and 4 of Def. 5.)*

*Proof.* For succeeding substitution by senders  $\{S_{i_1}, \dots, S_{i_k}\}$ , adversaries need to produce a fraudulent message  $\{b, m', h'\}$  such that  $h' = f(b)m' + g(b)$ ,  $b \notin \{b_{i_1}, \dots, b_{i_k}\}$  and  $\{b, m', h'\} \neq \alpha (= \{b_{i_0}, m, h\})$ , where  $\alpha$  is an authenticated message generated by  $S_{i_0}$ . For the fraudulent message  $\{b, m', h'\}$ , we consider the following cases: 1)  $b = b_{i_0}$  and  $m' \neq m$ , 2)  $b \neq b_{i_0}$  and  $m' = m$ , 3)  $b \neq b_{i_0}$  and  $m' \neq m$ . For case 1)  $b = b_{i_0}$  and  $m' \neq m$ , we have  $h' = f(b)(m' - m) + h$ . Since the adversaries only have  $f(b_{i_1}), \dots, f(b_{i_k})$ , and  $\deg f(x) = k + 1$ , the only information the adversaries have is  $f(b) \notin \{f(b_{i_1}), \dots, f(b_{i_k})\}$ . Consequently, there are  $q - k$  possible values for  $f(b)$ . Hence, from  $h' = f(b)(m' - m) + h$ , there also exist  $q - k$  different values for  $h'$  for any  $\{b_{i_0}, m, m', h\}$ . This implies that the probability for succeeding substitution does not exceed  $1/(q - k)$ . For case 2)  $b \neq b_{i_0}$  and  $m' = m$ , we have  $\deg(f(x)m' + g(x)) = k + 1$  and the adversaries have only  $f(b_{i_0})m' + g(b_{i_0})$  and  $f(b_{i_1})m' + g(b_{i_1}), \dots, f(b_{i_k})m' + g(b_{i_k})$ . Hence, the adversaries have no information on  $h = f(b)m' + g(b)$ , consequently, the probability for succeeding substitution also does not exceed  $1/q$ . For case 3)  $b \neq b_{i_0}$  and  $m' \neq m$ , we have  $\deg g(x) = k + 1$  and the adversaries only

have  $f(b_{i_0})m + g(b_{i_0})$  and  $g(b_{i_1}), \dots, g(b_{i_k})$ . This means, the adversaries have no information on  $g(b)$ . Also, this implies that they do not have any information on  $h'$  because  $h' = f(b)m' + g(b)$ , consequently, the probability for succeeding substitution also does not exceed  $1/q$ . Similarly, we can also prove that the probability of succeeding outsiders' substitution will be at most  $1/q$ .  $\square$  Together, any adversaries can succeed their impersonation or substitution with probability at most  $1/(q - k)$ .

**Lemma 6.**  *$R$  or  $G$  can determine who had generated the authenticated message  $\alpha$  with probability at most  $\Pr(\mathcal{S} = S_{i_0})$ . Additionally, cooperating with  $G$ ,  $R$  can reveal the identity of the sender of the authenticated message  $\alpha$  with probability 1, if  $\alpha$  is valid. (See conditions 5, 6 and 7 of Def. 5.)*

*Proof.* Regarding the sender's anonymity, it is clear that  $R$  has no information on the identity of the sender since  $R$  does not know the mapping  $\pi$ . Hence,  $R$  can only determine the probability of the generator of the authenticated message  $\alpha$  to be at most  $\Pr(\mathcal{S} = S_{i_0})$ . On the other hand, though  $G$  knows  $\pi$ ,  $G$  too, has no information regarding the identity of the sender since  $G$  does not know  $g(b_{i_0})$ . The probability of  $G$  determining who the generator of  $\alpha$  is, is at most  $\Pr(\mathcal{S} = S_{i_0})$ . However, by cooperating with  $G$ ,  $R$  can identify the sender with probability 1 if  $\alpha$  is valid.  $\square$

**Theorem 8.** *The above scheme is a  $(\frac{1}{q-k}, k, n)$ -one-time GA-code.*

*Proof.* From Lemmas 4, 5 and 6, it is obvious that the above theorem is true.  $\square$

In the security definition of GA-code, it is assumed that  $G$  does not join any colluders who try to perform impersonation or substitution. We should note that the probability of succeeding substitution can be increased when  $G$  joins a collusion attack. Since  $G$  knows  $f(b_i)$  which is assigned to  $S_i$ , for example, he can substitute a valid authenticated message  $\alpha := \{m, b_i, f(b_i)m + g(b_i)\}$  with a forged message  $\alpha' := \{m + 1, b_i, f(b_i)(m + 1) + g(b_i)\}$  which will be accepted by  $R$ . If such an attack is to be avoided, we can fix the above scheme with a slight modification as follows: TI uniformly at random chooses two mappings  $\pi_1 : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$  and  $\pi_2 : \{1, 2, \dots, n\} \rightarrow S$  such that  $\pi_2(\pi_1(b_i)) = S_i$  instead of  $\pi$ . Then,  $\{f(x), g(x), \pi_1\}$  and  $\pi_2$  are given to  $R$  and  $G$  as  $v$  and  $w$ , respectively.

The required memory sizes for the above construction is formally addressed as follows:

**Theorem 9.** *The required memory sizes in the above scheme are given as follows:*

$$H(\mathcal{A}) = \log_2 nq(q - 1), \quad H(\mathcal{U}_i) = \log_2 nq^2 \quad \text{for any } i \ (1 \leq i \leq n),$$

$$H(\mathcal{V}) = 2(k + 2) \log_2 q, \quad H(\mathcal{W}) = \sum_{i=0}^{n-1} \log_2 (q - i).$$

**CONSTRUCTION FROM COVER FREE FAMILY.** Another construction of GA-code is based on CFF. An advantage to use the CFF based GA-code is recalling USAE,

it does not always require  $|M| + 1 \geq n$  while the requirement is an absolute for the polynomial based GA-code.

In order to construct a GA-code from CFF, we also introduce “classical” A-codes [18,27] which include only one sender and one receiver. In such A-codes, there are 3 participants, a sender  $\tilde{S}$ , a receiver  $\tilde{R}$  and a trusted initializer  $\tilde{\text{TI}}$ .  $\tilde{\text{TI}}$  generates secret information  $\tilde{u}$  and  $\tilde{v}$  for  $\tilde{R}$  and  $\tilde{S}$ , respectively, such that  $\tilde{u} = \tilde{v}$ . In order to send a plaintext  $\tilde{m}$  to  $\tilde{R}$ ,  $\tilde{S}$  generates his authenticated message  $\tilde{\alpha}$  from  $\tilde{m}$  by using  $\tilde{u}$  and transmits  $\tilde{\alpha}$  to  $\tilde{R}$ .  $\tilde{R}$  verifies the validity of  $\tilde{\alpha}$  by using  $\tilde{m}$  and  $\tilde{v}$ . We note that  $\tilde{S}$  or  $\tilde{R}$  may generate  $\tilde{u}$  and  $\tilde{v}$  in order to remove  $\tilde{\text{TI}}$ .

**Definition 6.** Let  $(\tilde{U}, (\tilde{V}), \tilde{M})$  and  $(\tilde{A})$  denote the random variables induced by  $\tilde{u}, (\tilde{v}), \tilde{m}$  and  $\tilde{\alpha}$ , respectively. We say that  $(\tilde{U}, (\tilde{V}), \tilde{M}, \tilde{A})$  is a *p-authentication code* (A-code) if

1. Any outsiders (which do not include  $\tilde{S}$ ,  $\tilde{R}$  or  $\tilde{\text{TI}}$ ) can perform impersonation with probability at most  $p$ . Namely,  $\max_{\tilde{\alpha}} \Pr(\tilde{R} \text{ accepts } \tilde{\alpha}) \leq p$ .
2. Any set of outsiders can perform substitution with probability at most  $p$ . Namely, letting  $\tilde{\alpha}'$  be an authenticated message which is generated by  $\tilde{S}$ ,  $\max_{\tilde{\alpha}'} \max_{\tilde{\alpha}, \tilde{\alpha} \neq \tilde{\alpha}'} \Pr(\tilde{R} \text{ accepts } \tilde{\alpha} | \tilde{\alpha}') \leq p$ .

Construction methods of A-codes are given in, for example, [18,27]. In the followings, for simplicity, let  $f : \tilde{M} \times \tilde{U} \rightarrow \tilde{A}$  denote a mapping such that  $f(\tilde{m}, \tilde{u}) = \tilde{\alpha}$ . Additionally, notations for CFF is the same as that in Def. 3.

### GA-Code Based on Cover Free Families

**1. Setting Up:** Let  $M := \tilde{M}$ . TI first generates an  $(n, t, k)$ -CFF such that each of  $\ell_i$  ( $1 \leq i \leq t$ ) is an element of  $\tilde{U}$ . TI also chooses distinct numbers  $r_i$  ( $1 \leq i \leq n$ ) from  $\{1, 2, \dots, n\}$  uniformly at random. An algorithm that generates  $F_i$  ( $1 \leq i \leq n$ ) from  $i$  and  $L$  may be public to all players. TI further uniformly at random chooses two mappings  $\pi_1 : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$  and  $\pi_2 : \{1, 2, \dots, n\} \rightarrow S$  such that  $\pi_2(\pi_1(r_i)) = S_i$  for  $1 \leq i \leq n$ . Next, TI gives  $\{L, \pi_1\}$  to  $R$  as  $v$ . TI also gives  $\{r_i, F_{r_i}\}$  ( $1 \leq i \leq n$ ) to  $S_i$  ( $1 \leq i \leq n$ ), respectively, as  $u_i$ . In addition,  $\pi_2$  is given to  $G$  as  $w$ . After distributing the keys, TI deletes his memory.

**2. Message Generation:** Sender  $S_i$  generates an authenticated message  $\alpha$  for  $m$  as  $\alpha := \{r_i, \alpha'_1{}^{(r_i)}, \alpha'_2{}^{(r_i)}, \dots, \alpha'_{|F_{r_i}|}{}^{(r_i)}\}$ , where  $\alpha'_j{}^{(r_i)} := f(m, \ell_j^{(r_i)})$  ( $1 \leq j \leq |F_{r_i}|$ ), assuming that  $F_{r_i} = \{\ell_1^{(r_i)}, \ell_2^{(r_i)}, \dots, \ell_{|F_{r_i}|}^{(r_i)}\}$ .

**3. Verification:** Receiver  $R$  first generates  $F_{r_i}$  from  $L$  and  $r_i$ . Then,  $R$  accepts  $\alpha$  as valid if  $\alpha'_j{}^{(r_i)}$  is identical to  $f(m, \ell_j^{(r_i)})$  for all  $j$  ( $1 \leq j \leq |F_{r_i}|$ ).

**4. Tracing:** When  $R$  wants to reveal the identity of the sender,  $R$  first sends a request to  $G$ . If  $R$ 's request is approved by  $G$ ,  $R$  calculates  $t = \pi_1(r_i)$  and transmits it to  $G$ . Then,  $G$  reveals the sender's identity by  $S_i = \pi_2(t)$  and transmits this result back to  $R$  via a secure channel.

**Theorem 10.** *The above scheme is a  $(p, k, n)$ -one-time GA-code.*

The proof of the theorem is straightforward.

The required memory sizes for the above construction is formally addressed as follows:

**Theorem 11.** *The required memory sizes in the above scheme are given as follows:*

$$H(\mathcal{A}) = \log_2 n + |F|H(\tilde{\mathcal{A}}), \quad H(\mathcal{U}_i) = \log_2 n + |F|H(\tilde{\mathcal{U}}) \quad \text{for any } i \ (1 \leq i \leq n),$$

$$H(\mathcal{V}) = tH(\tilde{\mathcal{U}}) + \sum_{i=0}^{n-1} \log_2 (i+1), \quad H(\mathcal{W}) = \sum_{i=0}^{n-1} \log_2 (i+1),$$

assuming that all  $|F_{r_i}|$  ( $1 \leq i \leq n$ ) are of the same size  $|F|$ .

As mentioned so far, we see that the above scheme does not always require  $|M| + 1 \geq n$  while the polynomial based GA-code can be utilized only when  $|M| + 1 \geq n$ . In addition, it should be noticed that the size of  $\alpha$  can be reduced if each of  $\alpha_1^{(r_i)}, \alpha_2^{(r_i)}, \dots, \alpha_{|F_{r_i}|}^{(r_i)}$  contains the same  $m$ .

### 3.4 Remarks

In the previous subsection, we showed GA-codes in a single-receiver model. A multiple-receiver extension that was made similarly to MUSAE for GA-code was omitted here, but will appear later in the full version. Tight bounds for the required memory sizes in GA-code is important in analyzing optimality, and is also an interesting open problem to be thought out.

By the combination of USAE and GA-code, a secure communication system with confidentiality, authenticity and sender's anonymity was constructed. It should be noticed that the security of this system was proven without any computational assumptions and assures long-term security.

## References

1. M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-servers," Proc. of EUROCRYPT'98, LNCS 1403, Springer-Verlag, pp.437-447, 1998.
2. M. Bellare, A. Desai, E. Jorjipii and P. Rogaway, "A concrete security treatment of symmetric encryption," Proc. of 38th IEEE Symposium on Foundations of Computer Science (FOCS), pp.394-403, 1997.
3. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations among notions of security for public-key encryption schemes," Proc. of CRYPTO'98, LNCS 1462, Springer-Verlag, pp.26-45, 1998.
4. M. Ben-Or, S. Goldwasser and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," Proc. of 20th ACM Symposium on the Theory of Computing (STOC), pp.1-10, 1988.
5. R. Blom, "Non-public key distribution," Proc. of CRYPTO'82, Plenum Press, pp.231-236, 1983.
6. C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly secure key distribution for dynamic conferences," Proc. of CRYPTO'92, LNCS 740, Springer-Verlag, pp.471-486, 1993.

7. C. Blundo, L. A. Frota Mattos and D.R. Stinson, "Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution," Proc. of CRYPTO'96, LNCS 1109, Springer-Verlag, pp.387-400, 1996.
8. J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," Proc. of CRYPTO'97, LNCS 1294, Springer-Verlag, pp.410-424, 1997.
9. D. Chaum, "Untraceable electronic mail, return address, and digital pseudonyms," Communication of the ACM, 24, pp.84-88, 1981.
10. D. Chaum, "The dining cryptographers problem: unconditional sender and recipient untraceability," Journal of Cryptology, 1, 1, pp.65-75, 1987.
11. D. Chaum and E. van Heyst, "Group signatures," Proc. of EUROCRYPT'91, LNCS 547, Springer-Verlag, pp.257-265, 1991.
12. Y. Desmedt, Y. Frankel and M. Yung, "Multi-receiver/Multi-sender network security: efficient authenticated multicast/feedback," Proc. of IEEE Infocom'92, pp.2045-2054, 1992.
13. W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Trans. on Inform. Theory, IT-22, pp. 644-654, 1976.
14. D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography," Proc. of 23rd ACM Symposium on the Theory of Computing (STOC), pp.542-552, 1991.
15. P. Erdős, P. Frankl and Z. Füredi, "Families of finite sets in which no sets is covered by the union of two others," Journal of Combin. Theory Ser. A 33, pp.158-166, 1982.
16. P. Erdős, P. Frankl and Z. Füredi, "Families of finite sets in which no sets is covered by the union of  $r$  others," Israel Journal of Math., 51, pp.79-89, 1985.
17. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. on Inform. Theory, IT-31, 4, pp.469-472, 1985.
18. E. N. Gilbert, F. J. MacWilliams and N. J. A. Sloane, "Codes which detect deception," Bell System Technical Journal, 53, pp.405-425, 1974.
19. G. Hanaoka, J. Shikata, Y. Zheng and H. Imai, "Unconditionally secure digital signature schemes admitting transferability," Proc. of ASIACRYPT 2000, LNCS 1976, Springer-Verlag, pp.130-142, 2000.
20. G. Hanaoka, J. Shikata, Y. Zheng and H. Imai, "Efficient and unconditionally secure digital signatures and a security analysis of a multireceiver authentication code," Proc. of PKC 2002, LNCS 2274, Springer-Verlag, pp.64-79, 2002.
21. K. Kurosawa, T. Yoshida, Y. Desmedt and M. Burmester, "Some bounds and a construction for secure broadcast encryption," Proc. of ASIACRYPT'98, LNCS 1514, Springer-Verlag, pp.420-433, 1998.
22. T. Matsumoto and H. Imai, "On the KEY PREDISTRIBUTION SYSTEM: a practical solution to the key distribution problem," Proc. of CRYPTO'87, LNCS 293, Springer-Verlag, pp.185-193, 1987.
23. R. Rivest, "Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer," unpublished manuscript.
24. R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signature and public-key cryptosystems," Communication of the ACM, 21, 2, pp.120-126, 1978.
25. C. E. Shannon, "Communication theory of secrecy systems," Bell System Technical Journal, vol. 28, pp.656-715, 1949.
26. J. Shikata, G. Hanaoka, Y. Zheng and H. Imai, "Security notions for unconditionally secure signature schemes," Proc. of EUROCRYPT 2002, LNCS2332, Springer-Verlag, pp.434-449, 2002.

27. G. J. Simmons, "Authentication theory/coding theory," Proc. of CRYPTO'84, LNCS 196, Springer-Verlag, pp.411-431, 1984.
28. G. J. Simmons, "Message authentication with arbitration of transmitter/receiver disputes," Proc. of EUROCRYPT'87, Springer-Verlag, pp.151-165, 1987.
29. D. R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption," Designs, Codes and Cryptography, 12, pp.215-243, 1997.

# Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model

Alexander W. Dent\*

Information Security Group, Royal Holloway, University of London,  
Egham Hill, Egham, Surrey, UK,  
alex@fermat.ma.rhul.ac.uk, <http://www.isg.rhul.ac.uk/~alex/>

**Abstract.** The generic group model has recently been used to prove the security of certain asymmetric encryption and signature schemes. This paper presents results that show that there exist problems in that are provably hard in the generic group model but easy to solve whenever the random encoding function is replaced with a specific encoding function (or one drawn from a specific set of encoding functions). In particular we show that there exist cryptographic schemes that are provably hard in the generic group model but easy to break in practice.

## 1 Introduction

The complex nature of asymmetric encryption schemes makes it difficult to give concrete assurances of their security. In order to prove results about their security several models have been proposed. Each model makes some assumptions about the properties of certain parts of the scheme.

The most popular of these is the random oracle model, which was introduced by Bellare and Rogaway in 1993 [1]. It was designed to show the difficulty of breaking cryptographic algorithms by modelling certain parts of the cipher (usually the hash functions) as random functions. Doubt was cast on the validity of this model by Canetti, Goldreich and Halevi [3] who proved that there exists a theoretical signature scheme that is secure in the random oracle model but insecure when the random function is replaced by any polynomial time computable function or set of functions.

The generic group model was proposed by Shoup [8] to give exact bounds on the difficulty of the discrete logarithm problem and the Diffie-Hellman problem in the situation where the attacker has no information about the specific representation of the group being used. In other words the attacker is trying to solve a discrete logarithm (or Diffie-Hellman) problem in a group isomorphic to  $C_p$  but does not know whether this group is realised as, say, a multiplicative group or as an elliptic curve group. We cast some doubt on the model by proposing a problem that is provably difficult in the generic group model but for which

---

\* The work described in this paper has been supported by the Commission of the European Communities through the IST program under contract IST-1999-12324.



there exists an attacker that can easily solve the problem for any representation of the group without using any properties of the special properties of that representation.

More recently the generic group model has been used by Brown [2], Schnorr and Jakobsson [7], and Smart [9] in the analysis of certain cryptographic protocols based on the Diffie-Hellman problem. Our result shows that, in the analysis of asymmetric primitives, the generic group model has the same weaknesses as the random oracle model. In particular we show how a secure signature scheme may be modified to give a scheme that is still secure in the generic group model but insecure whenever any specific representation of the group is chosen.

This work is similar in its intent to the work of Fischlin [4] but our result is an improvement. Fischlin shows that the security of the Schnorr signature scheme [6] in the generic group model might depend upon the choice of hash function used within the scheme. The paper shows that the scheme is weak in the generic group model with one particular hash function and postulates that the scheme is secure in the generic group model with a different hash function. We improve upon this result and show that if there exists any signature scheme that is secure in the generic group model then there exists a tweaked version of that scheme that is still provably secure in the generic group model but insecure in practice.

## 2 The Generic Group Model

Let  $p$  be a  $k$ -bit prime and let  $\mathbb{Z}_p$  be the group of additive integers modulo  $p$ . Let  $l_{\text{out}} : \mathbb{N} \rightarrow \mathbb{N}$  be a length function with  $l_{\text{out}}(k) \geq k$  and  $S = \{0, 1\}^{l_{\text{out}}(k)}$ . Note that it is possible to represent elements of  $\mathbb{Z}_p$  as members of  $S$ . An encoding function is a function  $\sigma : \mathbb{Z}_p \rightarrow S$  for which  $\sigma(x) = \sigma(y)$  if and only if  $x = y$ .

The most common examples of encoding functions include representing an element  $x \in \mathbb{Z}_p$  as:

- the bit representation of  $x$  in  $\mathbb{Z}_p$ ,
- the bit representation of  $g^x$  in  $\mathbb{Z}_m$ , where  $g$  has order  $p$  in  $\mathbb{Z}_m$ ,
- the bit representations of the co-ordinates of the elliptic curve point  $xP$ , where  $P$  is a point of order  $p$  on an elliptic curve  $E$ .

It is important to note that finding  $x$  given  $\sigma(x)$  and  $\sigma(1)$  is the same as solving the discrete logarithm problem on the group.

A generic algorithm is a probabilistic, polynomial-time Turing machine  $M$  that takes representations of group elements  $\sigma(x_1), \dots, \sigma(x_m)$  as inputs. As  $M$  is executed it may compute group operations on group elements by way of an addition oracle  $\mathcal{O} : S \times S \times \mathbb{Z}_2 \rightarrow S$  such that

$$\mathcal{O}(\sigma(x_i), \sigma(x_j), b) = \sigma(x_i + (-1)^b x_j) . \quad (1)$$

We assume that any call to this oracle involves one evaluation of  $\sigma$ .

We will denote a generic algorithm  $M$  with access to an encoding function  $\sigma$  and a suitable addition/subtraction oracle by  $M^\sigma$  (we implicitly assume the

presence of an addition oracle whenever we have an oracle for the encoding function). We define the result of running such an algorithm as  $x \leftarrow M^\sigma$ . This differs from the original definition of [8] as our generic algorithm can calculate  $\sigma(x)$  for any  $x \in \mathbb{Z}_p$  without calculating any intermediate values. In particular  $M$  can calculate  $\sigma(1)$ .

This does not substantially change any of the results given in [8] because even when it is not possible to calculate  $\sigma(x)$  directly, it is always possible to calculate  $\sigma(x)$  using a polynomial number of queries to the addition oracle  $\mathcal{O}$ . The following is a result of Shoup [8].

**Result 1.** *Let  $x \in \mathbb{Z}_p$  and let  $\sigma$  be a randomly chosen encoding function of  $\mathbb{Z}_p$  into  $S$ . If  $M^\sigma$  is a generic algorithm for  $\mathbb{Z}_p$  on  $S$  that makes at most  $m$  evaluations of  $\sigma$  then the probability that  $x \leftarrow M^\sigma(\sigma(x))$  is  $O(m^2/p)$ . Note that in particular if  $M^\sigma$  makes a number of evaluations of  $\sigma$  that is polynomial in  $k$  then the probability that  $x \leftarrow M^\sigma(\sigma(x))$  is negligible.*

### 3 Evasive Relations on Groups

The definition of an evasive relationship was introduced in [3] and we will continue to develop definitions and use proof techniques that that paper suggests. The notion of evasive relations capture one difference between random functions (a function chosen at random from all possible functions) and functions actually used in practice (that must be calculatable).

**Definition 1 (Evasive Relation).** *A relation  $R \subseteq \{0, 1\}^* \times \{0, 1\}^{l_{\text{out}}(k)}$  is said to be evasive if for any probabilistic polynomial-time Turing machine  $M$  with access to an oracle  $\mathcal{P}$  we have*

$$\Pr[x \leftarrow M^{\mathcal{P}}(1^k), (x, \mathcal{P}(x)) \in R]$$

*is negligible in  $k$ , where the probability is taken uniformly over all choices of oracle  $\mathcal{P} : \{0, 1\}^* \rightarrow \{0, 1\}^{l_{\text{out}}(k)}$  and the coins of  $M$ .*

We extend this definition so that it is applicable to the group setting.

**Definition 2 (Evasive Group Relation).** *A relation  $R \subseteq G \times S$  is said to be an evasive group relation if for any probabilistic polynomial-time Turing machine  $M$  we have*

$$\Pr[x \leftarrow M^\sigma(1^k), (x, \sigma(x)) \in R]$$

*is negligible in  $k$ , where the probability is taken uniformly over all choices for an encoding function  $\sigma : G \rightarrow S$  and the coins of  $M$ .*

However, in the real world we will not be working with a random encoding function but with a known computable function that is, at worst, chosen from some collection. For example we could be working in a subgroup of the multiplicative group of integers modulo a value or a subgroup of an elliptic curve group with the points represented as either compressed, uncompressed or hybrid bit-strings. We designate the collection of these possible encoding functions an “encoding ensemble”.

**Definition 3 (Encoding ensemble).** We define an encoding ensemble  $\mathcal{F}$  to be a collection of encoding functions  $f_s : \mathbb{Z}_p \rightarrow S$  where  $s \in \{0, 1\}^k$ . (We do not require that  $\mathcal{F}$  contain exactly  $2^k$  functions, just that this is an upper bound). We require that there exists a polynomial-time algorithm EVAL such that  $\text{EVAL}(s, x) = f_s(x)$  and a polynomial-time algorithm ADD such that

$$\text{ADD}(s, f_s(x_i), f_s(x_j), b) = f_s(x_i + (-1)^b x_j) . \quad (2)$$

Once again we reiterate the fact that complete knowledge of an encoding function  $f_s$  and the encoding a group element  $f_s(x)$  does not imply that it is feasible to calculate  $x$ . This is the discrete logarithm problem and in general this is hard. However, in general, it is not necessary to be able to invert an encoding function in order to construct the ADD function - all of the examples of encoding functions given in Section 2 have efficient ADD functions, even when the discrete logarithm problem is thought to be hard for that representation.

We try to emulate the idea of evasive group relation when the randomly chosen encoding function is replaced with a function chosen at random from an encoding ensemble.

**Definition 4 (Correlation intractability).** Let  $\mathcal{F}$  be an encoding ensemble of  $\mathbb{Z}_p$  into  $S$ .  $\mathcal{F}$  is correlation intractable if for every probabilistic, polynomial-time Turing machine  $M$  and every evasive group relation  $R$  we have that

$$\Pr[s \leftarrow \{0, 1\}^k, x \leftarrow M(s), (x, f_s(x)) \in R]$$

is negligible in  $k$ , where the probability is taken over the uniformly random choice of  $s$  and the coins of  $M$ .

A clear example of the difference between random encoding functions (an encoding function drawn at random from all possible encoding functions) and encoding ensembles (where the encoding function is drawn from a specific set) is that there exists no encoding ensemble which is correlation intractable.

**Lemma 1.** *There exist no correlation intractable encoding ensembles.*

*Proof.* Let  $\mathcal{F}$  be an encoding ensemble of  $\mathbb{Z}_p$  into  $S$  and define the relation  $R$  to be

$$R = \{(\bar{s}, f_s(\bar{s})) : s \in \{0, 1\}^k\} \quad (3)$$

where  $\bar{s} = s \pmod{p}$ . This is an evasive relation because for every  $x \in \mathbb{Z}_p$  there exists at most two  $y$  such that  $(x, y) \in R$  and so, for any  $x \in \mathbb{Z}_p$ , we have that

$$\Pr[(x, \sigma(x)) \in R] \leq \frac{1}{2^{l_{\text{out}}(k)-1}} \leq \frac{1}{2^{k-1}} \quad (4)$$

for a randomly chosen encoding function  $\sigma$ .

However if  $M(s)$  is the machine that returns  $\bar{s}$  then

$$\Pr[s \leftarrow \{0, 1\}^k, \bar{s} \leftarrow M(s), (\bar{s}, f_s(\bar{s})) \in R] = 1 \quad (5)$$

for any random choice of  $s \in \{0, 1\}^k$ . So  $\mathcal{F}$  is not a correlation intractable encoding ensemble.  $\square$

## 4 A Hard Problem with an Easy Solution

In this section we will examine a slightly modified version of the discrete logarithm problem. We still attempt to solve the discrete logarithm problem in the group  $\mathbb{Z}_p$  as it is represented by the encoding function, however we now allow the attacking machine to have access to certain oracles. We attempt to show that whilst this problem is secure in the generic group model, it is insecure whenever any specific encoding function or encoding ensemble is used.

### 4.1 A Modified Problem

For any evasive group relation  $R$  we define an oracle  $D_R^\sigma$  such that

$$D_R^\sigma(y, \sigma(x)) = \begin{cases} x & \text{if } (y, \sigma(y)) \in R, \\ \perp & \text{otherwise.} \end{cases} \quad (6)$$

We still have that

**Theorem 2.** *If  $M^{\sigma, D_R^\sigma}$  is a generic algorithm that makes at most a number of queries to any oracle that is polynomial in  $n$  then*

$$\Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x))]$$

*is negligible, where the probability is taken over the uniform choice of encoding function  $\sigma$  and the coins of  $M$ .*

*Proof.* Obviously the oracle  $D_R^\sigma$  does not affect  $M$  unless it is queried with a value  $y$  such that  $(y, \sigma(y)) \in R$ . Since  $R$  is a group evasive relation this probability is negligible, hence we may ignore the oracle  $D_R^\sigma$ . However in this case we may appeal to Result [1](#), which proves that the probability of  $M^\sigma$  returning  $x$  without the oracle  $D_R^\sigma$  is also negligible.

Formally we define  $E$  to be the event that the oracle  $D_R^\sigma$  is queried with  $(y, z)$  such that  $(y, \sigma(y)) \in R$  and  $\bar{E}$  be the complement of this event. So,

$$\begin{aligned} \Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x))] &= \Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x)) | E] \Pr[E] \\ &\quad + \Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x)) | \bar{E}] \Pr[\bar{E}] \\ &\leq \Pr[E] + \Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x)) | \bar{E}] \end{aligned} \quad (7)$$

and both of these terms are negligible, the latter by Result [1](#).  $\square$

This proves that the oracle  $D_R^\sigma$  has no effect on the problem in the generic group model. Now consider the effects of this oracle when the random encoding function  $\sigma$  is replaced by an encoding ensemble. (Or rather the function  $\sigma$  chosen at random from all encoding functions is replaced with a function  $f_s$  chosen at random from the encoding ensemble  $\mathcal{F}$ .) If we use the group evasive relation  $R$  defined in [\(3\)](#) then the previously useless oracle  $D_R^\sigma$  now becomes

$$D_R^s(y, f_s(x)) = \begin{cases} x & \text{if } (y, f_s(y)) \in R, \\ \perp & \text{otherwise.} \end{cases} \quad (8)$$

Of course now there exists a machine  $M^{D_R^s}(f_s(x), s)$  that will output  $x$  with probability 1 just by querying the oracle  $D_R^s$  with the query  $(\bar{s}, f_s(x))$ , where  $\bar{s} \in \mathbb{Z}_p$  and  $\bar{s} \equiv s \pmod{p}$ .

## 4.2 A Universal Encoding Ensemble

So far we have shown that for every encoding ensemble there exists an oracle discrete logarithm problem that is provably difficult in the generic group model but easy when the random encoding function is replaced by a specific given encoding ensemble. We will now attempt to generalize this to an oracle discrete logarithm problem that is hard in the generic group model but easy when the random encoding function is replaced by *any* encoding ensemble. In order to do this we will need to enumerate all possible encoding ensembles.

Recall that for any function ensemble  $\mathcal{F}$  there exists a polynomial-time function  $\text{EVAL}(s, x)$  that evaluates  $f_s(x)$ . We cannot enumerate all polynomial-time functions as there is no single polynomial-time bound that they all obey, so instead we enumerate all functions that run in time  $t(k) = k^{\log k}$ . We do this by enumerating all algorithms and modifying each algorithm to force it to terminate after  $t(k)$  steps. Note that this enumeration will include all polynomial-time algorithms.

We denote the  $i$ -th encoding ensemble in this enumeration by  $\mathcal{F}^i$  and the  $s$ -th member of that encoding ensemble by  $f_s^i$ . We let  $\mathcal{U}$  denote the universal encoding ensemble given by

$$\mathcal{U}(\langle i, s \rangle, x) = f_s^i(x) \quad (9)$$

We remark that there exists a machine that computes  $\mathcal{U}$  and runs in time  $t(k)$ . Now consider the relation  $R$  induced by  $\mathcal{U}$  given by

$$R = \{(x, y) : y = \mathcal{U}(x, \bar{x})\} \subseteq \{0, 1\}^* \times S \quad (10)$$

where  $\bar{x}$  is the element of  $\mathbb{Z}_p$  such that  $\bar{x} \equiv x \pmod{p}$  (i.e.  $(x, y) \in R$  if and only if  $x = \langle i, s \rangle$  and  $y = f_s^i(\bar{x})$ ). This relation is clearly evasive as for any  $x$  there exists at most one value of  $y$  such that  $(x, y) \in R$ . Again we consider the oracle  $D_R^\sigma$  such that

$$D_R^\sigma(y, \sigma(x)) = \begin{cases} x & \text{if } (y, \sigma(y)) \in R, \\ \perp & \text{otherwise.} \end{cases} \quad (11)$$

Now we may deduce the following two results in exactly the same way as before but using the evasive relation  $R$  (which is slightly different to the evasive group relation we used before).

**Lemma 2.** *If  $M^{\sigma, D_R^\sigma}$  is a generic algorithm that make a polynomial number of queries to any oracle then*

$$\Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x))]$$

*is negligible, where the probability is taken over the uniform choice of encoding function  $\sigma$  and the coins of  $M$ .*

Now we replace the random encoding function  $\sigma$  with the label describing the encoding function,  $\langle i, s \rangle$ . It is important to replace  $\sigma$  with  $\langle i, s \rangle$  exactly. Since  $\sigma$  is an oracle that is available to both the attacker and the oracle  $D_R$  we must make sure that both the attacker and the oracle have access to a legitimate copy of  $\langle i, s \rangle$ . It is easiest to think of  $\langle i, s \rangle$  as a system parameter.

**Lemma 3.** *There exists a Turing machine  $M$  that runs in time polynomial in  $k$  such that*

$$\Pr[x \leftarrow M^{D_R^{(i,s)}}(f_s^i(x), \langle i, s \rangle)] = 1 \quad .$$

*Proof.*  $M$  queries  $D_R^{(i,s)}$  with the input  $(\langle i, s \rangle, f_s^i(x))$  and then outputs the output of the oracle. This can be done in polynomial time since we know  $f_s^i$  is a polynomial time encoding function.  $\square$

## 5 Signature Schemes

The results in this paper have been phrased in terms of an oracle problem that is provably hard in the generic group model. Some readers might dislike the use of a very powerful oracle that only outputs useful information in a very small number of cases. We have chosen to exhibit the results in the more general sense of a problem but it should also be noted that the above results could have been phrased in terms of a signature or encryption scheme. Here the oracle is replaced by access to a signing oracle or a decryption oracle, which seems much more natural.

### 5.1 A Signature Scheme That Runs in Super-polynomial Time

Suppose  $(\mathcal{S}, \mathcal{V})$  is a signature scheme secure against adaptive chosen message attacks in the generic group model. We can modify the scheme so that it is still secure in the generic group model but insecure when any encoding ensemble is used instead of a random encoding function. Let  $\mathcal{S}_1$  be the signing function given by

$$\mathcal{S}_1^\sigma(m, sk) = \begin{cases} sk \parallel \mathcal{S}^\sigma(m, sk) & \text{if } (m, \sigma(m)) \in R, \\ \mathcal{S}^\sigma(m, sk) & \text{if } (m, \sigma(m)) \notin R. \end{cases} \quad (12)$$

where  $m$  is the message to be signed,  $sk$  is the secret key and  $R$  is the relation given in equation [10](#). The corresponding verifying algorithm,  $\mathcal{V}_1$  is given by

$$\mathcal{V}_1^\sigma(m, s, pk) = \begin{cases} \mathcal{V}^\sigma(m, s', pk) & \text{if } (m, \sigma(m)) \in R \text{ and } s = x \parallel s' \\ & \text{(where } x \text{ is the same length as } sk), \\ \mathcal{V}^\sigma(m, s, pk) & \text{if } (m, \sigma(m)) \notin R. \end{cases} \quad (13)$$

where  $m$  is the message,  $s$  is the signature and  $pk$  is the public key. The signature scheme  $(\mathcal{S}_1, \mathcal{V}_1)$  is still secure against adaptive chosen message attacks in the generic group model.

However we have already shown that once we replace the random encoding function  $\sigma$  with an encoding function  $f_s$  drawn at random from an encoding ensemble  $\mathcal{F}^i$  then we can find a message  $m = \langle i, s \rangle$  for which  $(m, f_s^i(m)) \in R$ . Hence we completely recover the secret key if we query the signing oracle with  $m$ . So the scheme is insecure for any concrete instantiation of the encoding function, i.e. the scheme is insecure in practice.

Unfortunately we are not quite finished: at the moment both the signing and verifying algorithms run in time  $t(k) = O(k^{\log k})$ . This is because both algorithms need to check a relation in  $R$  and then only way to check if  $(\langle i', s' \rangle, y) \in R$  is to check if  $f_{s'}^{i'}(\langle i', s' \rangle) = y$ , which may take super-polynomial time.

## 5.2 Running the Scheme in Polynomial Time

We will use the CS-proof techniques of Micali [5] to run this scheme in polynomial time. Unlike [3] we cannot use guaranteed CS-proofs as we are unable to easily construct independent random functions<sup>1</sup>, so we will instead use the notion of a *cryptographic CS-proof*. For this we require that all parties have access to a common random string  $r$ . Micali [5] shows that there exists polynomial-time algorithms PRO and VER such that

- if  $(x, \sigma(x)) \in R$  then PRO that computes a proof  $\pi$  to this fact,
- if  $(x, \sigma(x)) \in R$  and  $\pi$  is a proof to this fact then VER verifies this proof,
- if  $(x, \sigma(x)) \notin R$  then a polynomial-time adversary produces a proof  $\pi'$  that VER accepts for only an exponentially small number of random strings  $r$ .

From the details of [5] we see that it is reasonable to assume that the last fact goes even further: for any random string  $r$  it is computationally infeasible for a polynomial-time adversary to find a group element  $x$  and a proof  $\pi'$  such that  $(x, \sigma(x)) \notin R$  but VER accepts the proof.

So we may now define a new signature scheme  $(\mathcal{S}_2, \mathcal{V}_2)$  that is still secure against adaptive chosen ciphertext attacks in the generic group model. Note that any message  $m$  may be written as  $x||\pi$  where  $x$  is a group element, hence we may define the signing function on a message  $m$  to be:

$$\mathcal{S}_2^\sigma(m, sk) = \begin{cases} sk||\mathcal{S}^\sigma(m, sk) & \text{if VER verifies the proof } \pi \text{ that} \\ & (x, \sigma(x)) \in R, \\ \mathcal{S}^\sigma(m, sk) & \text{otherwise.} \end{cases} \quad (14)$$

where  $sk$  is the secret key. The corresponding verifying function for a message  $m$  and a proposed signature  $s$  is given by:

$$\mathcal{V}_2^\sigma(m, s, pk) = \begin{cases} \mathcal{V}^\sigma(m, s', pk) & \text{if VER verifies the proof } \pi \text{ that} \\ & (x, \sigma(x)) \in R \text{ and } s = x||s' \text{ (where} \\ & x \text{ is the same length as } sk), \\ \mathcal{V}^\sigma(m, s, pk) & \text{otherwise.} \end{cases} \quad (15)$$

This scheme is secure in the generic group model because it is computationally infeasible to guess  $x$  such that  $(x, \sigma(x)) \in R$  and it is also computationally infeasible to produce a proof  $\pi$  that will fool the signing oracle into believing that  $(x, \sigma(x)) \in R$ . Furthermore, since VER runs in polynomial-time, both the signing and verifying functions run in polynomial-time.

<sup>1</sup> Of course, we could allow all parties to have access to a random oracle and then use the construction given in [3]. This would then allow us to prove that, in the random oracle model, there exists a scheme that is secure in the generic group model but insecure in any practical situation. Alternatively we could construct a scheme that is secure in the combined random oracle/generic group model but insecure in the standard model (i.e. when all random functions are replaced with functions drawn from the relevant ensembles). Whilst this technique is used successfully in Schnorr and Jakobsson [7] we feel that, in this particular situation, this is too much like passing the buck!

However when the random encoding function is replaced by the encoding function  $f_s^i$  then an attacker could submit the message

$$m = \langle i, s \rangle \parallel \text{PRO}(x, r) \quad (16)$$

to the signing oracle and the signing oracle will return the secret key. So the scheme is insecure for any practical instantiation of the encoding function.

## 6 Conclusion

We have shown that the generic group model suffers from the same weaknesses as the random oracle model, namely, that a problem can be shown to be hard in the generic group model but is easy when the random function is changed to any specific function or set of functions. This shows that the generic group model is not a perfect way to represent an algorithm that attacks a problem defined on a group but doesn't take advantage of any of the specific group structure.

We have also adapted this to show that there are cryptographic schemes that are secure in the generic group model that are insecure whenever a specific encoding function is used. Heuristically this means that security proofs that rely on the generic group model should be viewed with the same caution as security proofs that rely on the random oracle model.

## Acknowledgements

The author would like to thank the Information Security Group of Royal Holloway and, in particular, Kenny Paterson and Christine Swart for helpful discussions on this subject.

## References

1. M. Bellare and P. Rogaway. 'Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.' *Proceedings of the First ACM Conference on Computer and Communications Security*, 1993.
2. D. Brown. 'Generic Groups, Collision Resistance, and ECDSA.' Available from <http://eprint.iacr.org/>, 2002.
3. R. Canetti, O. Goldreich and S. Halevi. 'The Random Oracle Methodology, Revisited.' *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, 1998.
4. M. Fischlin. 'A Note on Security Proofs in the Generic Model.' *Advances in Cryptology - Asiacrypt 2000*, 2000.
5. S. Micali. 'CS proofs.' *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, 1994.
6. C. Schnorr. 'Efficient Signature Generation for Smart Cards.' *Journal of Cryptology*, Vol 4, 1991.
7. C. Schnorr and M. Jakobsson. 'Security of Signed El-Gamal Encryption.' *Advances in Cryptology - Asiacrypt 2000*, 2000.



8. V. Shoup. 'Lower Bounds for Discrete Logarithms and Related Problems.' *Theory and Application of Cryptographic Techniques*, 1997.
9. N. Smart. 'The Exact Security of ECIES in the Generic Group Model.' *Cryptography and Coding*, 2001.

# On the Impossibilities of Basing One-Way Permutations on Central Cryptographic Primitives

Yan-Cheng Chang<sup>1</sup>, Chun-Yun Hsiao<sup>1</sup>, and Chi-Jen Lu<sup>2</sup>

<sup>1</sup> Department of Computer Science and Information Engineering,  
National Taiwan University, Taipei, Taiwan,  
{r88023,r88067}@csie.ntu.edu.tw

<sup>2</sup> Institute of Information Science,  
Academia Sinica, Taipei, Taiwan,  
cjl@iis.sinica.edu.tw

**Abstract.** We know that trapdoor permutations can be used to construct all kinds of basic cryptographic primitives, including trapdoor functions, public-key encryption, private information retrieval, oblivious transfer, key agreement, and those known to be equivalent to one-way functions such as digital signature, private-key encryption, bit commitment, pseudo-random generator and pseudo-random functions. On the other hand, trapdoor functions are not as powerful as trapdoor permutations, so the structural property of permutations seem to be something special that deserves a more careful study. In this paper, we investigate the relationships between one-way permutations and all these basic cryptographic primitives. Following previous work, we focus on an important type of reductions called black-box reductions. We prove that no such reductions exist from one-way permutations to either trapdoor functions or private information retrieval. Together with previous results, all the relationships with one-way permutations have now been established, and we know that no such reductions exist from one-way permutations to any of these primitives except trapdoor permutations. This may have the following meaning, with respect to black-box reductions. We know that one-way permutations imply none of the primitives in “public cryptography”, where additional properties are required on top of “one-wayness” [12], so permutations cannot be traded for any of these additional properties. On the other hand, we now know that none of these additional properties can be traded for permutations either. Thus, permutation seems to be something orthogonal to those additional properties on top of one-wayness. Like previous non-reducibility results [12, 23, 17, 7, 9, 8, 6], our proofs follow the oracle separation paradigm of Impagliazzo and Rudich [12].

## 1 Introduction

Modern cryptography has provided us with all kinds of protocols for various interesting and important tasks involving security issues. However, almost all of

these protocols have their securities based on some intractability assumptions which all imply  $\mathcal{P} \neq \mathcal{NP}$ . So unconditional proofs of security for these protocols may seem far beyond our reach. One important line of research then is to understand the relationships among these assumptions. However, there are many interesting cryptographic tasks, and even a single task may be several variants. So potentially the whole picture could become very messy and have little help in clarifying our understanding. Instead, we want to focus on the most basic cryptographic tasks in their most primitive forms, which can serve as building blocks for more advanced protocols. We will also restrict ourselves to the classical world of cryptography, and leave the questions in quantum cryptography for future studies.

According to [7], such basic cryptographic primitives can be roughly divided into two categories: private cryptography and public cryptography.<sup>1</sup> Private cryptography is represented by private-key encryption, and includes one-way permutation (OWP), one-way function (OWF), pseudo-random generator (PRG), pseudo-random functions (PRF), bit commitment (BC), and digital signature (DS). Public cryptography is represented by public-key encryption (PKE), and includes trapdoor permutations (TDP), trapdoor functions (TDF), oblivious transfer (OT), private information retrieval (PIR), and key agreement (KA). “One-wayness” turns out to be essential as these primitives all are known to imply one-way functions [11,19,1,2,7]. For private cryptography, one-wayness basically is also sufficient as one-way functions can be used to construct all the primitives therein, except one-way permutations. For public cryptography, additional properties are required on top of one-wayness, and the relationships among primitives appear to be rather complicated. We know that trapdoor permutations imply all of them, but some implications among others are known to fail, in the sense to be discussed next.

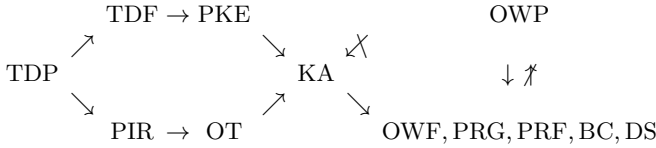
It is not clear what it means that one primitive  $Q$  does not imply the other primitive  $P$ , or equivalently  $P$  can not be reduced to  $Q$ , especially when both primitives exist under some plausible assumptions. After all, if the primitive  $P$  exists, there is a protocol of  $P$  based on  $Q$  that simply ignores  $Q$ . Impagliazzo and Rudich [12] introduced a restricted but important subclass of reductions called *black-box reductions*. Informally speaking, a black-box reduction from  $P$  to  $Q$  is a construction of  $P$  out of  $Q$  that ignores the internal structure of the implementation of  $Q$ . Furthermore, the security of  $P$ ’s implementation can also be guaranteed in a black-box way that one can use any adversary breaking  $P$  as a subroutine to break  $Q$ . In fact in cryptography, almost all constructions of one primitive from another known so far are done in this way, so it makes sense to focus on reductions of this kind. Hereafter, all the reductions or implications we refer to in this paper will be black-box ones. To prove that no black-box reduction exists from  $P$  to  $Q$ , it suffices to construct an oracle relative to which  $Q$  exists

---

<sup>1</sup> We want to remark that this classification is just a convenient one for us and is by no means a precise or complete one. The situation becomes complicated when one wants to talk about variations of primitives meeting additional requirements (e.g. [23,6]).

whereas  $P$  does not. Using this approach, Impagliazzo and Rudich [12] showed that no such black-box reduction exists from KA to OWP. As every primitive in public cryptography implies KA [14, 7], this provides a strong evidence that primitives in public cryptography requires strictly more than one-wayness. Since then, more and more separations between cryptographic primitives have been established following this paradigm [21, 23, 17, 7, 9, 8, 6].

We know that trapdoor permutations imply all those basic cryptographic primitives, but it is not the case for trapdoor functions as they do not imply OT [7] and thus PIR [4]. So there seems to be something special for being a permutation which deserves further study. We also know that one-way functions do not imply one-way permutations [20, 16], so permutation does not seem to be a property that one can have for free. We know that one-way permutations imply none of the primitives in public cryptography [12], so on top of one-wayness, one can not trade permutations for any of the additional properties required in public cryptography. Then, the question we want to ask is: can any of those additional properties required in public cryptography be traded for permutations? Formally, can any of the primitives except TDP in public cryptography imply OWP? Figure 1 summarizes the relationships known so far between primitives and OWP. We will show that neither TDF nor PIR implies OWP, so the answer to that question is actually no!



**Fig. 1.** Relationships between OWP and other cryptographic primitives

We first construct an oracle, relative to which an injective trapdoor function (iTDF) exists whereas OWP does not. As iTDF implies PKE [24]<sup>2</sup> and PKE (two-pass KA) implies KA, we establish the impossibility of having black-box reductions from OWP to either TDF, PKE, or KA. Next, we construct an oracle, relative to which PIR exists whereas OWP does not. Because PIR implies OT [4], we establish that no black-box reduction exists from OWP to either PIR or OT. One immediate corollary is that PIR does not imply TDP, in contrast to the known result that TDP does imply PIR [15]. So according to our results, none of the primitives in public cryptography implies OWP in a black-box way. This is interesting in the sense that all the powerful primitives, except TDP, in public cryptography, which make almost all of conceivable cryptographic tasks possible, are still unable to yield OWP. Our results suggest that permutation is really a special property that is orthogonal to other additional properties required in cryptography. Furthermore, the reducibility from OWP

<sup>2</sup> In fact, TDF with polynomial pre-image size suffices to imply PKE [1].

to each primitive was already known before, so now all the relationships, with respect to black-box reductions, between one-way permutations and those basic cryptographic primitives have been established. However, we want to stress that we are still far from being able to settle the real relationships among primitives, and in fact, to have separations beyond black-box reductions would require some major breakthrough in complexity theory [12].

For each separation between primitives, we need to find a suitable oracle that is powerful enough for making one primitive possible, but still not so for the other. We basically follow the approach of Impagliazzo and Rudich [12] and Gertner *et al.* [7]. It is known that a random function is one-way with high probability, even relative to a  $\mathcal{PSPACE}$ -complete function [12]. Then, OWF exists relative to an oracle containing a random function and a  $\mathcal{PSPACE}$ -complete function, but on the other hand, OWP does not relative to such an oracle [20,16]. We want to separate OWP from TDF and PIR. Each time we look for a special function which realizes the additional property required by that primitive but does not yield permutations. By adding such a function to the oracle, we can build the corresponding primitive, TDF or PIR, but relative to the oracle, OWP still does not exist. Our strategy of finding such special functions is based on the observation that both TDF and PIR can be seen as two-party primitives while OWP involves only one party. So we look for those functions that are useful in a two-party setting but useless in a one-party case.

The rest of the paper is organized as follows. In Section 2, we describe our notation and provide definitions for the cryptographic primitive involved in this paper. Then in Section 3 and 4, we prove that no black-box reductions exist from OWP to iTDF and PIR, respectively.

## 2 Notation and Definitions

Let  $[n]$  denote the set  $\{0, 1, \dots, n-1\}$ . For  $x \in \{0, 1\}^n$ , let  $x[i]$  denote the  $i$ -th bit of  $x$  if  $i \in [n]$ , and an arbitrary value, say 0, otherwise. We write  $\text{poly}(n)$  to denote a polynomial in  $n$ . We write  $*$  for  $\{0, 1\}^*$  and  $(*, q, *)$  for those  $(u, q, v)$  with  $u, v \in \{0, 1\}^*$ . For a distribution  $S$ , we write  $s \in S$  to denote sampling  $s$  according to the distribution  $S$ . For any  $n \in \mathbb{N}$ , let  $U_n$  denote the uniform distribution over  $\{0, 1\}^n$ .

Parties in cryptographic primitives are assumed to run in polynomial time, and are modeled by probabilistic polynomial-time Turing machines (PPTM). Each cryptographic primitive is associated with a security parameter  $k$ , for evaluating how secure that primitive is. A function is called negligible if it vanishes faster than any inverse polynomial. We say that two distributions  $X$  and  $Y$  over  $\{0, 1\}^k$  cannot be distinguished if for any PPTM  $M$ ,

$$\left| \Pr_{x \in X}[M(x) = 1] - \Pr_{y \in Y}[M(y) = 1] \right| \leq \delta(k),$$

for some negligible function  $\delta(k)$ . We say a function is easy to compute if it is computable in polynomial time. We say that a function  $f$  is hard to invert if for

any PPTM  $M$ ,

$$\Pr_{x \in U_k} [f(M(f(x))) = f(x)] \leq \delta(k),$$

for some negligible function  $\delta(k)$ .

In the following, we give brief definitions of the cryptographic primitives studied in this paper. More formal treatment can be found in standard textbooks or the original papers. The most fundamental primitive is *one-way function*, which is essential to all cryptographic primitives.

**Definition 1.** *A one-way function (OWF) is a function that is easy to compute but hard to invert.*

From one-way functions, we define primitives with additional properties. A one-way permutation is a one-way function that is itself a permutation.

**Definition 2.** *A one-way permutation (OWP) is a one-way function  $f$  with the additional requirement that for every  $k \in \mathbb{N}$ ,  $f$  maps  $\{0, 1\}^k$  to  $\{0, 1\}^k$  in a one-to-one and onto way.*

Trapdoor functions are one-way functions which, when given some additional *trapdoor* information, are easy to invert.

**Definition 3.** *A collection of trapdoor functions (TDF) is a collection of function families  $\mathcal{F} = \{\mathcal{F}_k | k \in \mathbb{N}\}$  satisfying the following properties.*

- *There is a PPTM  $I$ , that on input  $1^k$  outputs a pair  $(f, t)$ , where  $f$  is (an index of) a function in  $\mathcal{F}_k$  and  $t$  is a string called the trapdoor for  $f$ .*
- *Each  $f$  is easy to compute, and when the trapdoor  $t$  is given,  $f$  is also easy to invert.*
- *For a random  $(f, t) \in I(1^k)$ ,  $f$  is hard to invert without knowing the trapdoor  $t$ .*

Next, we describe private information retrieval, which was introduced by Chor *et al.* [3]. This is a two-party protocol, where User wants to secretly learn some bit of Server's database, conditioned on a non-trivial upper bound on Server's communication complexity.

**Definition 4.** *Private information retrieval (PIR) is a protocol involving two parties. Server has a database  $x \in \{0, 1\}^n$  while User has an index  $i \in [n]$  and wants to learn the bit  $x[i]$  in the following way.*

- *Server sends less than  $n$  bits to User.*
- *User keeps the index secret in the sense that Server cannot distinguish the distributions of messages sent from User when the indices are  $i$  and  $i'$  respectively, for any  $i' \neq i$  [3].*

---

<sup>3</sup> The security parameter here can be set to  $k = \text{poly}(n)$ .

### 3 TDF Does Not Imply OWP

In this section we construct an oracle  $\Gamma$  relative to which there are injective trapdoor functions but no one-way permutations. It is shown in [20,16] that no OWP exists relative to an oracle with a  $\mathcal{PSPAC}\mathcal{E}$ -complete problem and some random functions. We add a function  $G$  into such an oracle to do the inverting job when provided with the trapdoor, and we want  $G$  to be useless in constructing OWP. Our oracle  $\Gamma$  consists of the following.

- A  $\mathcal{PSPAC}\mathcal{E}$ -complete problem.
- A length-tripling random function  $F(\cdot, \cdot)$ .
- A length-tripling random function  $H(\cdot)$ .
- A function  $G$  defined as follows.

$$\forall(u, v) : \quad G(u, v) = \begin{cases} w & \text{if } \exists w : u = F(w, H(v)), \\ \perp & \text{otherwise.} \end{cases}$$

In  $\Gamma$ , the functions  $F$  and  $H$  are random while the function  $G$  is completely determined by  $F$  and  $H$ . Call a query to  $G$  *invalid* if its answer is  $\perp$ , and *valid* otherwise. Note that we can assume w.l.o.g. that both  $F$  and  $H$  are injective, because one can show that length-tripling functions are injective on sufficiently long inputs with measure one.

$G$  is designed in this way for the following purpose. The function  $F(\cdot, H(t))$  can be inverted if one has  $t$ , because for any  $x$ ,

$$G(F(x, H(t)), t) = x.$$

Without knowing  $t$ , queries to  $G$  are likely to be invalid and thus useless. As we will see, this makes the construction of trapdoor functions possible. On the other hand, the function  $G$  is not helpful in a one-party primitive (OWP in particular), for the following reason. To have a valid query  $G(y, t)$ ,  $y$  is likely to come from a query  $F(x, H(t))$  for some  $x$ , but then one knows  $x = G(y, t)$  already, which makes such a query to  $G$  unnecessary. Our approach basically follows those of [12,7].

#### 3.1 TDF in $\Gamma$

On input  $1^k$ , the trapdoor-function generator  $I$  outputs the pair  $(t, H(t))$ , where  $t \in U_k$  is the trapdoor and  $H(t)$  is the index for the function  $F(\cdot, H(t))$ . For convenience, we write  $F_t(\cdot)$  to denote the function  $F(\cdot, H(t))$ , and assume its domain being  $\{0, 1\}^k$ . Given the index  $H(t)$ , the function  $F_t$  is easy to compute, just by querying the oracle  $F(\cdot, H(t))$ . Having the trapdoor  $t$ ,  $F_t$  is easy to invert, with the help from the oracle  $G$  as

$$\begin{aligned} G(F_t(x), t) &= G(F(x, H(t)), t) \\ &= x. \end{aligned}$$

It remains to show that  $F_t$  is hard to invert without knowing the trapdoor  $t$ .

Consider any oracle PPTM  $M$  as an inverter. Without the oracle  $G$ ,  $F_t$  is a random function and is likely to be one-way, by a standard argument (e.g. [20]). The idea is that unless  $M^\Gamma$  can guess the trapdoor  $t$  correctly,  $G$  is unlikely to provide useful information for inverting  $F_t$ . Formally, for a negligible function  $\delta(k)$ , we want to upper-bound the probability

$$\Pr_{\Gamma} \left[ \Pr_{x,t} [M^\Gamma(F_t(x), H(t)) = x] > \delta(k) \right],$$

which by Markov inequality is at most

$$\mathbb{E}_F \left[ \Pr_{x,t} [M^\Gamma(F_t(x), H(t)) = x] \right] / \delta(k) = \Pr_{\Gamma, x, t} [M^\Gamma(F_t(x), H(t)) = x] / \delta(k).$$

We need the following lemma.

**Lemma 1.**  $\Pr_{\Gamma, x, t} [M^\Gamma(F_t(x), H(t)) = x] \leq k^c 2^{-k}$ , for some constant  $c$ .

*Proof.* Define the following probability event:

- $B_1$ :  $M^\Gamma$  on input  $(F_t(x), H(t))$  queries  $H$  on  $t$  or  $G$  on  $(*, t)$ .

We first show that this bad event is unlikely to happen.

*Claim.*  $\Pr_{\Gamma, x, t} [B_1] \leq \text{poly}(k) 2^{-k}$ .

*Proof.* Note that whether or not  $M^\Gamma$  queries  $H$  on  $t$  or  $G$  on  $(*, t)$  does *not* depend on either  $H(t)$  or  $G(*, t)$ . Instead, it is completely determined by the input together with those  $H(t')$  and  $G(*, t')$  for every  $t' \neq t$ . Fix any  $x, t$  and any restriction  $\Gamma_0$  of  $\Gamma$  that leaves only  $H(t)$  random. Note that  $G(*, t)$  is not fixed yet as it depends on  $H(t)$ , but it has no effect on  $B_1$ . Then whether or not  $B_1$  happens depends only on the input, because all oracle answers that may matter have been fixed. Therefore,

$$\begin{aligned} \Pr_{\Gamma, x, t} [B_1] &= \mathbb{E}_{x, t, \Gamma_0} \left[ \Pr_{H(t)} [M^{\Gamma_0}(F(x, H(t)), H(t)) \text{ queries } H \text{ on } t \text{ or } G \text{ on } (*, t)] \right] \\ &= \mathbb{E}_{x, t, \Gamma} \left[ \Pr_h [M^\Gamma(F(x, h), h) \text{ queries } H \text{ on } t \text{ or } G \text{ on } (*, t)] \right] \\ &= \mathbb{E}_{x, \Gamma, h} \left[ \Pr_t [M^\Gamma(F(x, h), h) \text{ queries } H \text{ on } t \text{ or } G \text{ on } (*, t)] \right] \\ &\leq \text{poly}(k) 2^{-k}, \end{aligned}$$

where the last inequality is because  $M$  makes at most  $\text{poly}(k)$  queries.  $\square$

Next, we want to show that if the bad event  $B_1$  does not happen,  $M^\Gamma$  is unlikely to invert the input correctly. We may assume w.o.l.g. that  $M^\Gamma$  always uses its output to query  $F_t$  at the final step before it stops. This does not affect its inverting probability, which is bounded above by the probability of the following event:



–  $B_2$ :  $M^\Gamma$  on input  $(F_t(x), H(t))$  queries  $F_t$  on  $x$ .

So it remains to prove the following claim.

*Claim.*  $\Pr_{\Gamma, x, t}[B_2 | \neg B_1] \leq \text{poly}(k)2^{-k}$ .

*Proof.* The proof is very similar to that of Claim [1](#) by observing the correspondence between  $(x, F_t)$  and  $(t, H)$ . Fix any  $x, t$  and any restriction  $\Gamma_1$  of  $\Gamma$  that leaves only  $F_t(x)$  random. Again  $G(*, t)$  is not determined yet but it has no effect as it is not queried conditioned on  $\neg B_1$ . Then whether or not  $M^\Gamma$  queries  $F_t$  on  $x$  is completely determined by the input, because all oracle answers that may matter have been fixed. The rest is similar.  $\square$

With these two claims, we have

$$\begin{aligned} \Pr_{\Gamma, x, t}[M^\Gamma(F_t(x), H(t)) = x] &\leq \Pr_{\Gamma, x, t}[B_1] + \Pr_{\Gamma, x, t}[B_2 | \neg B_1] \\ &\leq \text{poly}(k)2^{-k} + \text{poly}(k)2^{-k} \\ &\leq k^c 2^{-k}, \end{aligned}$$

for some constant  $c$ . This completes the proof of Lemma [1](#)  $\square$

Let  $\delta(k) = k^{c+2}2^{-k}$  and we have

$$\Pr_{\Gamma} \left[ \Pr_{x, t}[M^\Gamma(F_t(x), H(t)) = x] > \delta(k) \right] \leq \frac{1}{k^2}.$$

Now as  $\sum_k \frac{1}{k^2}$  converges, the *Borel-Cantelli Lemma* tells us that with probability one over  $\Gamma$ ,  $\Pr_{x, t}[M^\Gamma(F_t(x), H(t)) = x]$  is negligible for sufficiently large  $k$ . There are only countably many machines  $M$ 's, each of which can only succeed as an inverter over a measure zero of  $\Gamma$ , so we have the following.<sup>[4](#)</sup>

**Lemma 2.** *Relative to measure one of random  $\Gamma$ , injective trapdoor functions exist.*

### 3.2 No OWP in $\Gamma$

In this section we show that no OWP exists relative to  $\Gamma$ . It was shown in [\[20, 16\]](#) that no OWP exists relative to an oracle with a  $\mathcal{PSPACE}$ -complete problem and some random functions. We proceed by showing that the function  $G$  does not help us build OWP either. The idea is that it is unlikely to have a valid long input  $(F(x, H(t)), t)$  without querying  $F$  at  $(x, H(t))$  first. But with  $x$ , the answer to the query  $G(F(x, H(t)), t)$ , one can eliminate this application of  $G$ . We can see the random oracle  $\Gamma$  as a family of oracles, with each oracle in the family being a possible instance of  $\Gamma$ .

<sup>4</sup> Like previous work on this subject, we only consider uniform adversaries. The analysis does not appear to work against non-uniform adversaries, as there are uncountably many of them.

Assume for the contrary that OWP exists relative to  $\Gamma$ . According to [20], this implies that for any constant  $\delta > 0$ , there exists a machine  $M$  that computes OWP on measure  $1 - \delta$  of oracles in  $\Gamma$ . Let  $\Gamma'$  denote this subset of oracles relative to which  $M$  is a OWP. We will show that for this  $M$ , there is another machine  $N$  which never queries  $G$  but still produces the same outputs for most inputs. Then we will show that a good inverter exists for  $N$ , which can also invert  $M$  well, so  $M$  cannot be one-way.

Consider inputs from  $\{0, 1\}^n$ . Suppose  $M$ 's running time is bounded by  $n^c$ , for some constant  $c \geq 2$  independent of  $n$ . For this constant  $c$ , let  $N$  be the machine that simulates  $M$  step by step, keeps track of the queries to  $F$ , and answers any query to  $G$ , say on  $(u, v)$ , by the following. Look for  $w$  with  $u = F(w, H(v))$ , by going through previous queries to  $F$  or searching the space  $\{0, 1\}^{|u|/3}$  if  $|u| \leq 3c \log n$ . If such  $w$  is found,  $N$  answers  $G(u, v)$  with it. Otherwise  $N$  assumes  $(u, v)$  an invalid query and answers it with  $\perp$ . This takes at most polynomial time.

For any input  $x \in \{0, 1\}^n$ ,  $N(x) \neq M(x)$  only if  $M$  every queries  $G$  on some valid  $(u, v)$  with  $u$  longer than  $3c \log n$  but not obtained by previous queries to  $F$ . Then for any fixed random choice of  $M$ ,  $N(x) \neq M(x)$  for at most  $n^c 2^{c \log n} / 2^{3c \log n} = 1/n^c \leq 1/n^2$  of oracles in  $\Gamma$ , and hence for at most  $1/((1 - \delta)n^2) \leq 2/n^2$  of oracles in  $\Gamma'$ , for  $\delta \leq 1/2$ . Although we can then show that relative to most oracles  $M$  and  $N$  agree on most inputs, but  $N$  may not be a permutation relative to most oracles. So we can not apply [20] directly to invert  $N$ , and some modification is needed. First, we can have the following.

**Lemma 3.** *There are less than  $2/n$  fraction of  $n$ -bit strings  $y$  such that  $N^{-1}(y) \neq M^{-1}(y)$  for more than  $2/n$  of oracles in  $\Gamma'$ .*

*Proof.* Consider the Boolean matrix  $A$  with rows indexed by  $y \in \{0, 1\}^n$  and columns indexed  $\gamma \in \Gamma'$ , such that  $A_{y,\gamma} = 1$  iff  $N^{-1}(y) \neq M^{-1}(y)$  relative to  $\gamma$ . For each  $x \in \{0, 1\}^n$ ,  $N(x) \neq M(x)$  for at most  $2/n^2$  of oracles in  $\Gamma'$ , and this contributes at most  $2^{-n} 4/n^2$  fraction of 1's to  $A$ . As there are  $2^n$  different  $x$ 's, the total fraction of 1's in  $A$  is at most  $4/n^2$ . By the pigeon-hole principle, less than  $2/n$  of rows in  $A$  have more than  $2/n$  of columns of 1's.  $\square$

For any  $y$ ,  $M^{-1}(y)$  is unique relative to any oracle in  $\Gamma'$  since it is a permutation. So by Lemma 3, there are more than  $1 - 2/n$  fraction of  $n$ -bit strings  $y$  such that  $N^{-1}(y)$  is unique for more than  $1 - 2/n$  of oracles in  $\Gamma'$ , and hence for more than  $1 - 2/n - \delta > 1 - \epsilon$  of oracles in  $\Gamma$ , for any constant  $\epsilon > \delta$  and sufficiently large  $n$ . Observe that based on [16], the proofs of Theorem 9.2 and 9.3 in [20] actually yield the following stronger statement.

**Lemma 4.** *Assume  $\mathcal{P} = \mathcal{NP}$ . There is a constant  $\lambda$  such that for every machine  $N$ , there exists a machine  $N'$  with the following property. For any  $\epsilon < \lambda$  and for any  $y$ , if  $N^{-1}(y)$  is unique for  $1 - \epsilon$  of random oracles, then  $N'(y) = N^{-1}(y)$  for  $1 - \sqrt[\epsilon]{\epsilon}$  of random oracles.*

Then the rest follows closely the proof of Theorem 9.4 in [20]. Choose  $\delta < \lambda$  such that there exists  $\epsilon$  with  $\delta < \epsilon < \lambda$  and  $\epsilon + \sqrt[\epsilon]{\epsilon} < 1$ . We have  $\mathcal{P} = \mathcal{NP}$

relative to  $\Gamma$ , so for any  $n$ , there are more than  $1 - 2/n$  of  $n$ -bit string  $y$  such that we can find  $N^{-1}(y) = M^{-1}(y)$  for more than  $1 - \sqrt{\epsilon}$  of oracles in  $\Gamma$ . By the pigeon-hole principle, there are more than  $1 - \sqrt[4]{\epsilon}$  of oracles in  $\Gamma$  relative to which we can compute  $M^{-1}(y)$  for more than  $1 - 2/n - \sqrt[4]{\epsilon}$  fraction of  $n$ -bit strings  $y$  for infinitely many  $n$ . That is,  $M$  is one-way relative to less than  $\sqrt[4]{\epsilon} < 1 - \epsilon < 1 - \delta$  fraction of oracles in  $\Gamma$ , a contradiction. Thus, with probability one over  $\Gamma$ , no one-way permutation exists relative to  $\Gamma$ . Together with Lemma 2, we have the following theorem.

**Theorem 1.** *There is no black-box reduction from OWP to iTDF.*

## 4 PIR Does Not Imply OWP

In this section we construct an oracle  $\Phi$  relative to which PIR exists but OWP does not. Similar to section 3 we add a special function  $G$  to an oracle consisting of a  $\mathcal{PSPACE}$ -complete problem and some random functions. The oracle  $\Phi$  consists of the following.

- A  $\mathcal{PSPACE}$ -complete oracle.
- A length-tripling random function  $F(\cdot, \cdot)$ .
- A random function  $T : \{0, 1\}^* \rightarrow \{0, 1\}$ .
- A family of random functions  $H = \{H_k : \{0, 1\}^* \rightarrow \{0, 1\}^k | k \in \mathbb{N}\}$ .
- A family of functions  $G = \{G_k | k \in \mathbb{N}\}$  defined as follows.

$$\forall(u, v) : \quad G_k(u, v) = \begin{cases} u[s] \oplus T(H_k(u), t) & \text{if } \exists s, t : v = F(s, t), \\ \perp & \text{otherwise.} \end{cases}$$

The idea behind this design is the following. In PIR, User shall use  $F$  to encrypt her index  $i$  as  $F(i, m)$ , and Server shall call  $G$  with  $F(i, m)$  and his database  $x$  to get

$$G(x, F(i, m)) = x[i] \oplus T(H(x), m),$$

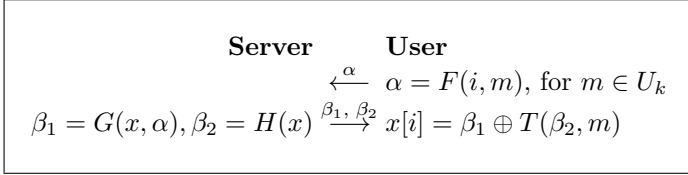
an encryption of  $x[i]$ , which can only be decrypted by User. As in the previous section, we will next show that the function  $G$  is not useful for a one-party primitive, and thus not useful for building OWP.

Although the oracle  $\Phi$  is designed to enable PIR, we stress that the definition of  $\Phi$  does not depend on any instance of PIR. In  $\Phi$ , the functions  $F, T, H$  are random, and the function  $G$  is completely determined by  $F, T, H$ . When we want to carry out a particular PIR instance, the oracle functions will then be queried at some particular places. For example, with database  $x$  and index  $i$ ,  $G$  will be queried at  $(x, F(i, m))$  for a random  $m$ .

Note that  $G$  is a family of functions, but later when we refer to it, we usually mean some  $G_k \in G$ , and similarly for  $H$ .  $G$  is well defined if  $F$  is injective, which is not an issue as with probability one, it is so for sufficiently long inputs, and we can make  $G$  outputs 0 on those short inputs. Call a query  $(u, v)$  to  $G$  *valid* if  $G(u, v) \neq \perp$ .

#### 4.1 PIR in $\Phi$

The following is a 2-pass PIR using the oracle  $\Phi$ , where Server has  $x \in \{0, 1\}^n$  and User has  $i \in [n]$ . Let  $k$  be the security parameter. For this parameter, we let  $H$  denote  $H_k$  and let  $G$  denote  $G_k$ .



The idea is the following. User needs to send her index  $i$  to Server in some way in order to obtain the bit  $x[i]$ . As User does not want Server to learn her index  $i$ , she would like to have it encrypted. So User chooses a random private key  $m$  and uses the random function  $F$  to encrypt  $i$  as  $F(i, m)$ . Server receives  $F(i, m)$  but has no idea about  $i$ . How can Server send information about  $x[i]$  to User without explicitly knowing the index  $i$ ? The function  $G$  does the magical work, which takes any  $x$  together with  $F(i, m)$  and returns the bit

$$G(x, F(i, m)) = x[i] \oplus T(H(x), m),$$

an encryption of  $x[i]$ . We want  $x[i]$  encrypted, since otherwise Server may recover  $i$  by calling  $G$  using several different  $x$ 's (User's security will be proved later). On the other hand, User has the key  $m$ , so after receiving  $G(x, F(i, m))$  and  $H(x)$ , she can query  $T(H(x), m)$  and derive

$$x[i] = G(x, F(i, m)) \oplus T(H(x), m).$$

The total number of bits sent by Server to User is

$$|\beta_1| + |\beta_2| = 1 + k,$$

which is okay when  $n > 1 + k$ .

It remains to prove User's security. Note that Server cannot affect what User would send, so whether Server is malicious or not makes no difference on User's security. If Server never queries the function  $G$ , the proof is standard as the rest of the oracle consists of merely random functions. The idea is that unless Server can guess User's private key  $m$  correctly, queries to  $G$  are unlikely to provide useful information. To see this, assume Server does not know  $m$ . The function  $H$  serves as a random hash and it is unlikely for Server to find distinct  $x', x''$  such that  $H(x') = H(x'')$  due to the large image of  $H(\cdot)$ , for sufficiently large  $k$ . Then for a query  $G(x', F(i, m))$ , the answer  $x'[i] \oplus T(H(x'), m)$  is likely to look random as  $T(H(x'), m)$  is likely so, and such a query is unlikely to be useful. That is, unless  $G$  is queried at  $(x', \alpha)$  and  $(x'', \alpha)$  for such  $x', x''$ ,  $G$  looks like a random function too.

Formally, we show that Server cannot distinguish the messages from User having indices  $i$  and  $j$  respectively. Consider any machine  $M$  as a distinguisher.

Let  $\delta(n) = 2^{-k/4}$ , a negligible function in  $n$ . For any  $i, j \in [n]$ , define  $\Delta_m^{i,j} = M^\Phi(F(i, m)) - M^\Phi(F(j, m))$ , which is a random variable of  $\Phi$ . Then

$$\mathbb{E}_m[\Delta_m^{i,j}] = \Pr_m[M^\Phi(F(i, m)) = 1] - \Pr_m[M^\Phi(F(j, m)) = 1].$$

We want to bound the probability

$$\begin{aligned} \Pr_\Phi \left[ \exists i, j : \left| \mathbb{E}_m[\Delta_m^{i,j}] \right| > \delta(n) \right] &\leq \sum_{i,j} \Pr_\Phi \left[ \left| \mathbb{E}_m[\Delta_m^{i,j}] \right| > \delta(n) \right] \\ &\leq \sum_{i,j} \mathbb{E}_\Phi \left[ \left( \mathbb{E}_m[\Delta_m^{i,j}] \right)^2 \right] / \delta^2(n). \end{aligned}$$

So we need the following lemma.

**Lemma 5.**  $\forall i, j, \mathbb{E}_{\Phi}[\mathbb{E}_m[(\Delta_m^{i,j})^2]] \leq \text{poly}(n)2^{-k}$ .

*Proof.* Fix any  $i, j \in [n]$ . Write  $\Delta_m$  for  $\Delta_m^{i,j}$  and note that  $\mathbb{E}_\Phi[(\mathbb{E}_m[\Delta_m])^2] = \mathbb{E}_{\Phi, m, m'}[\Delta_m \Delta_{m'}]$ . Define the following probability events, with  $\Phi, m, m'$  chosen randomly:

- $B_1$ : On input  $F(i, m)$  or  $F(j, m)$ ,  $M^\Phi$  queries on  $(*, m)$  or knows distinct  $x', x''$  with  $H(x') = H(x'')$ .
- $B_2$ : On input  $F(i, m')$  or  $F(j, m')$ ,  $M^\Phi$  queries either  $F(*, m)$ ,  $T(*, m)$ , or  $G(*, F(*, m))$ .

These are the bad events, which happen with probability at most  $\text{poly}(n)2^{-k}$ . Next we show that the expectation of  $\Delta_m \Delta_{m'}$  is small if neither bad event happens.

Consider any restriction  $\Phi_0$  of  $\Phi$  with  $F(*, m)$  and  $T(*, m)$  still random but the rest fixed.  $M$ 's computation is determined by the input and the answers to its oracle queries.

Assume the condition  $\neg B_1$ . Consider any possible run of  $M^{\Phi_0}(F(i, m))$  and  $M^{\Phi_0}(F(j, m))$ , starting with  $F(i, m) = F(j, m)$  and then getting same oracle answers, up to some query. Assume that now for some  $x'$ ,  $G(x', F(i, m))$  and  $G(x', F(j, m))$  are queried respectively, as other oracle answers are fixed under  $\Phi_0$ . The answers  $x'[i] \oplus T(H(x'), m)$  and  $x'[j] \oplus T(H(x'), m)$  have the same distribution as  $T(H(x'), m)$  remains free up to this point. By induction,  $M^{\Phi_0}(F(i, m))$  and  $M^{\Phi_0}(F(j, m))$  have the same distribution of computations. So given  $\neg B_1$ ,  $\mathbb{E}_{\Phi_0}[M^{\Phi_0}(F(i, m))] = \mathbb{E}_{\Phi_0}[M^{\Phi_0}(F(j, m))]$  and  $\mathbb{E}_{\Phi_0}[\Delta_m] = 0$ .

Consider any  $m' \neq m$ . Given  $\neg B_2$ ,  $\Delta_{m'}$  is fixed under  $\Phi_0$  as it does not depend on  $F(*, m)$  or  $T(*, m)$ . Let  $B = B_1 \cup B_2$ . Then given  $\neg B$ ,  $\mathbb{E}_{\Phi_0}[\Delta_m \Delta_{m'}] = \mathbb{E}_{\Phi_0}[\Delta_m] \Delta_{m'} = 0$  for any restriction  $\Phi_0$ . Thus,

$$\begin{aligned} \mathbb{E}_{\Phi, m, m'}[\Delta_m \Delta_{m'} | \neg B] &\leq \Pr_{m, m'}[m = m'] \\ &= 2^{-k}, \end{aligned}$$

and we have

$$\begin{aligned} \mathbb{E}_{\Phi, m, m'} [\Delta_m \Delta_{m'}] &\leq \Pr_{\Phi, m, m'} [B] + \mathbb{E}_{\Phi, m, m'} [\Delta_m \Delta_{m'} | \neg B] \\ &\leq \text{poly}(n) 2^{-k}. \end{aligned}$$

□

Then,  $\Pr_{\Phi} [\exists i, j | \mathbb{E}_m [\Delta_m^{i,j}] > \delta(n)] \leq \text{poly}(n) 2^{-k/2}$ . As  $\sum_n \text{poly}(n) 2^{-k/2}$  converges for, say,  $k = \Omega(\log^2 n)$ , the Borel-Cantelli Lemma tells us that with probability one over  $\Phi$ ,  $|\mathbb{E}_m [\Delta_m^{i,j}]| \leq \delta(n)$  for any  $i, j \in [n]$  for sufficiently large  $n$ . There are only countably many machines  $M$ 's as distinguishers, each of which succeeds with measure zero over  $\Phi$ . Then with probability one over  $\Phi$ , Server cannot learn User's index for sufficiently large  $n$ . So we have the following.

**Lemma 6.** *Our protocol is a PIR relative to measure one of  $\Phi$ .*

## 4.2 No OWP in $\Phi$

The proof that no OWP exists in  $\Phi$  is almost identical to the one in Section 3.2. Assume the contrary that there is a PPTM  $M$ , with time bound  $n^c$ , that computes a OWP. We construct  $N$  by simulating  $M$  and replacing any query to  $G$  at  $(u, v)$  by  $\perp$  if  $v$  is longer than  $3c \log n$  and not obtained from a previous query to  $F$ . If  $v$  is obtained from a previous query  $F(s, t)$  or short enough to find  $s, t$  by exhaustive search,  $N$  replace  $G(u, v)$  by  $u[s] \oplus T(H(u), t)$ . Then, as in Section 3.2  $N$  has the same output as  $M$  does on most inputs, but  $N$  can be inverted on most inputs. It follows that  $M$  is not one-way, a contradiction. So we have the following.

**Theorem 2.** *There is no black-box reduction from OWP to PIR.*

Together with Theorem 1 and previous results, we have the following.

**Corollary 1.** *There is no black-box reduction from OWP to any of the basic primitives, including TDF, PKE, PIR, OT, KA, OWF, PRG, PRF, BC, and DS.*

## References

1. Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 283–298. Springer-Verlag, 1998.
2. Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. One-way functions are essential for single-server private information retrieval. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 89–98, 1999.

3. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 41-50, 1995.
4. Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 122-138. Springer-Verlag, 2000.
5. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644-654, 1976.
6. Marc Fischlin. On the impossibility of constructing non-interactive statistically-secret protocols from any trapdoor one-way function. In Bart Preneel, editor, *Topics in Cryptology—CT-RSA '02*, volume 2271 of *Lecture Notes in Computer Science*, pages 79-95. Springer-Verlag, 2002.
7. Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 325-335, 2000.
8. Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 126-135, 2001.
9. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 305-313, 2000.
10. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364-1396, 1999.
11. Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 230-235, 1989.
12. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44-61, 1989.
13. Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 20-31, 1988.
14. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 364-373, 1997.
15. Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 104-121. Springer-Verlag, 2000.
16. Jeff Kahn, Michael E. Saks, and Cliff Smyth. A dual version of Reimer's inequality and a proof of Rudich's conjecture. In *Proceedings of the 15th Annual IEEE Conference on Computational Complexity*, pages 98-103, 2000.
17. Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 535-542, 1999.
18. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151-158, 1991.

19. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387-394, 1990.
20. Steven Rudich. Limits on the provable consequences of one-way functions. *Ph.D. thesis*, U.C. Berkeley, 1988.
21. Steven Rudich. The use of interaction in public cryptosystems (extended abstract). In Joan Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 242-251. Springer-Verlag, 1991.
22. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120-126, 1978.
23. Daniel R. Simon. Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology—EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334-345. Springer-Verlag, 1998.
24. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80-91, 1982.



# A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order

Ivan Damgård<sup>1</sup> and Eiichiro Fujisaki<sup>2</sup>

<sup>1</sup> BRICS, Dept. of Computer Science, Aarhus University,  
Ny Munkegade AARHUS C DK-8000 Denmark,  
ivan@daimi.au.dk

<sup>2</sup> NTT Labs, 1-1 Hikarinooka, Yokosuka-shi 239-0847 Japan,  
fujisaki@isl.ntt.co.jp

**Abstract.** We present a statistically-hiding commitment scheme allowing commitment to arbitrary size integers, based on any (Abelian) group with certain properties, most importantly, that it is hard for the committer to compute its order. We also give efficient zero-knowledge protocols for proving knowledge of the contents of commitments and for verifying multiplicative relations over the integers on committed values. The scheme can be seen as a generalization, with a slight modification, of the earlier scheme of Fujisaki and Okamoto [14]. The reasons we revisit the earlier scheme and give some modification to it are as follows:

- The earlier scheme [14] has some gaps in the proof of soundness of the associated protocols, one of which presents a non-trivial problem which, to the best of our knowledge, has remained open until now. We fill all the gaps here using additional ideas including minor modification of the form of a commitment.
- Although related works such as [8, 3, 10, 4] do not suffer from the main problem we solve here, the reason for this is that they use “commitments” with a single base (i.e., of form  $c = g^s \bmod n$ ). Such commitments, however, cannot satisfy the standard hiding property for commitments, and hence protocols using them cannot in general be (honest-verifier) zero-knowledge nor witness indistinguishable.
- In a computationally convincing proof of knowledge where the prover produces the common input (which is the type of protocol we look at here), one cannot completely exclude the possibility that a prover manages to produce a common input on which he can cheat easily. This means that the standard definition of proofs of knowledge cannot be satisfied. Therefore we introduce a new definition for computationally convincing proofs of knowledge, designed to handle the case where the common input is chosen by the (possibly cheating) prover.
- Our results apply to any group with suitable properties. In particular, they apply to a much larger class of RSA moduli than the safe prime products proposed in [14] – Potential examples include RSA moduli, class groups and, with a slight modification, even non-Abelian groups.

Our scheme can replace the earlier one in various other constructions, such as the efficient interval proofs of Boudot [4] and the efficient proofs for the product of two safe primes proposed by Camenisch and Michels [9].

# 1 Introduction

## 1.1 Statistically-Hiding Commitment and Associated Protocols

The notion of commitment is at the heart of many cryptographic protocols. The basic functionality one wants from a commitment is that the committer may choose in private a secret  $s$  from some set  $S$  and release some information, *the commitment* to a verifier, such that: even though the scheme is *hiding*, i.e., the verifier cannot compute anything about  $s$  from the commitment, it is also *binding*, i.e., the committer cannot change his mind after having committed, but he can later open the commitment to reveal  $s$ , and convince the verifier that this was indeed the original value committed to.

In many applications, one wants extra functionality from a commitment scheme, for instance that the committer can prove in zero-knowledge that he knows how to open a given commitment, in particular that he knows the value committed to. Also, if  $S$  has an algebraic structure, say as a ring or a group, it can be very useful to have a *multiplication protocol*, i.e., a zero-knowledge protocol in which the committer can prove that committed values  $a, b, c$  satisfy  $ab = c$ . If  $S$  is a ring, one can often, in addition, achieve that from commitments to  $a, b \in S$ , the verifier can compute a commitment to  $a + b$  without interacting with the committer.

One example of such a scheme where  $S = \mathbb{Z}/q\mathbb{Z}$ , where  $q$  is a prime, is the scheme of Pedersen [17]. For the associated protocols and additional examples, see [7]. In the vast majority of examples known, the set  $S$  is  $\mathbb{Z}/m\mathbb{Z}$  for some  $m$ , where  $m$  may or may not be a prime. A multiplication protocol for such a scheme is a protocol by which one can demonstrate that for committed numbers  $a, b, c$ ,  $ab = c \bmod m$  holds. However, there are several important cases where what you actually need is something stronger, namely to be able to prove that  $ab = c$  holds *over the integers*. One example of this is if you want to show that a committed number  $s$  is an RSA signature on a given message  $a$  w.r.t. public key  $n$ , 3. What we want to know is that  $a = s^3 + tn$  for some  $t$ , and this of course must be true over the integers and not just modulo  $m$ . Of course, one might be able to solve this by choosing the commitment scheme such that  $m = n$ , but this requires that at least you know  $n$  at the time the commitment scheme was set up, and also a new instance of the commitment scheme for each  $n$ . This is often unreasonable in practice. There are other ways around the problem, see for instance [12], but the protocols are far from optimal, typically one has to resort to “binary cut-and-choose”, which means communication complexity at least quadratic in the security parameter. Another example of the need for relations over the integers is the efficient zero-knowledge proofs of Boudot [4] for demonstrating that a committed number is in a given interval. Here, it is crucial for efficiency that one can prove efficiently that committed numbers  $a, b$  satisfy  $b = a^2$  over the integers.

It should be clear that what we really need here is an *integer* commitment scheme, that is, a scheme where  $S = \mathbb{Z}$  (or at least some large finite interval), and where there is an efficient multiplication protocol that works over the integers.

Here, by efficient, we mean constant round protocols requiring only communication linear in the security parameter.

## 1.2 The Earlier Scheme with Statistically-Hiding Commitment

In [14], Okamoto and the second author of this paper presented the first efficient integer commitment scheme and also suggested an efficient multiplication protocol. The scheme is based on the strong RSA assumption suggested in [2,14]. However, via private communication, we found some gaps in the proof of soundness of the associated protocols, one of which we think presents a non-trivial problem which, to the best of our knowledge, has remained open until now. Later in the paper we give a short explanation of the problem in the proof from [14]. We fill all the gaps here using additional idea including a minor modification of the form of a commitment.

## 1.3 Other Related Works

There are several related works inspired by [14] such as [8,3,10,4]. The protocols constructed there generally do not suffer from the main problem we mentioned above. However, the reason for this is that they use “commitments” with a single base, i.e., a commitment to  $s$  is of form  $c = g^s \bmod n$ . Such a commitment does not satisfy the standard hiding property for commitments. For instance, if a prover commits twice to the same value, this is immediately visible. Thus derived protocols using such commitments are not in general (honest-verifier) zero-knowledge nor witness indistinguishable.

Boudot [5] pointed out another problem in the proof of soundness – In this type of protocols (based on a group with a hidden order), the natural protocol for showing that one knows how to open a commitment  $c$  can in fact only show that the prover can open  $c$  or  $-c$  (a problem that even [8,3,10,4] cannot avoid). This is not so serious in practice but we suggest a solution to this problem too, by changing the way in which commitments are opened.

## 1.4 Our Scheme

In this paper, we present a commitment scheme that may be seen as a generalization of the Fujisaki-Okamoto scheme. We start with an arbitrary Abelian group  $G$ , with some basic properties. We assume that the verifier can choose the group and publish a *description* of it that allows anyone to compute the group and inversion operations in  $G$ . For the RSA case, this amounts to publishing the modulus  $n$ . The most important extra property we need is that it is hard, given the description, to extract the roots of a given random element in  $G$ . This is just a natural generalization of the strong RSA assumption. Some extra technical conditions are needed as well, we detail those later. We then build from this an integer commitment scheme, as well as a zero-knowledge protocol for proving knowledge of how to open a commitment, and an efficient zero-knowledge

multiplication protocol. In order to analyze these protocols, we introduce a new definition of computationally convincing proofs of knowledge, designed to handle the case where the common input is chosen by the (possibly cheating) prover. Our analysis is done in the exact security setting.

If we specialize to the case where  $G = (\mathbb{Z}/n\mathbb{Z})^\times$  for an RSA modulus  $n$ , we obtain - modulo some technical changes - the commitment scheme of Fujisaki and Okamoto, in particular we get what appears to be the first secure multiplication protocol for this type of scheme. In addition, the conditions we need on  $G$  turn out to translate into conditions on  $n$  that are much milder than those needed in the original paper [14], namely that  $n = pq$  is a safe prime product. We only need that  $\gcd(p-1, q-1) = 2$  and  $p-1, q-1$  don't have too many small prime factors (whose precise description follows below). Finally, our construction is applicable to groups other than RSA, for instance class groups. Here, it should be noted that finding roots in a class group seems to require finding the order of the group, and this problem is known to be at least as hard as factoring, and may in fact be harder.

Our commitment scheme and protocols are not exactly the same as those of [14], even when specialized to  $G = (\mathbb{Z}/n\mathbb{Z})^\times$ . However, with some minor technical changes of the commitment scheme in [14], one can give correct proofs for soundness of their protocols following the ideas we give here. However, our protocols are slightly more efficient than those of [14].

There are several variants for our protocols that we do not explain here due to space limitations, except for a few extensions given in Appendix B.

## 2 Model

As usual, probability  $\epsilon(k)$  will be called *negligible* if for all polynomials  $f(\cdot)$ , we have  $\epsilon(k) \leq 1/f(k)$  for all large enough  $k$ . On the other hand,  $1 - \epsilon(k)$  will be called *overwhelming* if  $\epsilon(k)$  is negligible. Also, we say  $\epsilon(k)$  is *significant* if for some polynomial  $f(k)$ , we have  $\epsilon(k) \geq 1/f(k)$  for all large enough  $k$ .

Suppose now that we are given a probabilistic polynomial time algorithm  $\mathcal{G}$  which on input  $1^k$  outputs a description  $\text{descr}(G)$  of a finite Abelian group  $G$ , where we assume one can efficiently verify from  $\text{descr}(G)$  that it actually specifies such a group. The algorithm may also output some side information, such as the order of  $G$ , or the prime factorization of the order; it may even be possible to ensure that the order of the group satisfies certain conditions. An example of such a  $\mathcal{G}$  is an RSA key generation algorithm - in this case it is indeed possible to generate a group with known and controlled order.

Given  $\text{descr}(G)$ , we assume that one can compute efficiently some estimates on the order,  $2^A \leq \text{ord}(G) \leq 2^B$ , where  $A$  and  $B$  are polynomial in  $k$ . We also assume that elements can be sampled randomly from the group and that inversion and group operation can be computed efficiently.

In order for our protocols to work, we need, loosely speaking, that it is hard to find non-trivial roots of elements in  $G$ . Furthermore, we need a condition on the structure of  $G$ 's output by  $\mathcal{G}$ . Loosely speaking, we need that  $G$  has a

large subgroup with only large prime factors in its order. To make this more precise, we assume that two functions are associated with  $\mathcal{G}$ :  $C(\cdot)$ ,  $l(\cdot)$ , that map positive integers to positive integers. Typically,  $C(k)$  is super-polynomially large as a function of  $k$ , whereas  $l(k)$  is always a polynomial. For any  $G$  produced by  $\mathcal{G}$  on input  $1^k$ , we will consider primes greater than  $C(k)$  as being “large”. By the structure theorem for Abelian groups, we can always write  $G = U \times H$ , where the order of  $H$  has only prime factors larger than  $C(k)$ , and the order of  $U$  has only prime factors at most  $C(k)$ . We say  $|H|$  is  $C(k)$ -rough, as opposed to being  $C(k)$ -smooth, which means a number has only prime factors less than  $C(k)$ . Thus  $l_G := |U|$  is  $C(k)$ -smooth.

We are now ready to state our assumptions about groups output by  $\mathcal{G}$ :

**Group Assumption.** For any  $G$  generated by  $\mathcal{G}$  on input  $1^k$  the following hold:

1. Write  $G = U \times H$  as above, with  $C(k)$ -smooth,  $l_G = |U|$  and  $C(k)$ -rough  $|H|$ . Then  $l_G \leq l(k)$  and  $\text{descr}(G)$  includes  $l_G$ .
2. For any string  $Y$ , when given  $\text{descr}(G)$ , it can be decided in (deterministic) polynomial-time in  $k$  whether  $Y$  represents an element in  $G$ .

**Root Assumption.** Let  $A$  be a probabilistic algorithm. We run  $\mathcal{G}$  on input  $1^k$  to get  $\text{descr}(G)$ . We give  $\text{descr}(G)$  and a random  $Y \in G$  as an input to  $A$ . We say that  $A$  solves the root problem if  $A$  outputs an integer  $e(> 1)$ ,  $X \in G$ , and  $\mu \in U$  such that  $Y = \mu X^e$  (where  $\mu \in U$  can be verified by checking that  $\mu^{l_G} = 1 \in G$ ). In particular, we say that the root problem is  $(t(k), \epsilon(k))$ -secure if for  $k$ , any adversary  $A$  that runs in time at most  $t(k)$ , solves the root problem with probability of at most  $\epsilon(k)$ . The probability is taken over the coin tosses of  $\mathcal{G}$  and  $A$ , as well as the random choice in  $G$ .

Some remarks on the assumptions:

The condition that  $l_G \leq l(k)$  says that  $G$  has many elements with only large prime factors in their orders: If  $Y$  is chosen randomly in  $G$ , then there is a significant probability,  $1/l(k)$ , that the order of  $Y$  is  $C(k)$ -rough. We want to stress that it is essentially important that membership in  $G$  can be decided efficiently – although this property is often ignored and forgotten, this was one reason why proofs of soundness for the earlier protocols suggested in [14] were incomplete. We will assume throughout that when a party receives an element that is supposed to be in  $G$ , membership in  $G$  is always checked.

The assumption that  $l_G$  is known and is part of the description can be removed, if in the protocols to follow, one replaces exponentiations to the power  $l_G$  by exponentiations to  $l(k)!$ . The price is loss of efficiency, since  $l(k)! \sim \sqrt{2\pi l(k)} l(k)^{l(k)+1/2} e^{-l(k)} \gg l_G$ . However, the cost to exponentiate to power  $l(k)!$  is still polynomial in  $k$ .

The second assumption is a generalization of the strong RSA assumption – we require that extracting non-trivial roots is hard, even if one is allowed to multiply the input by an element of relatively small known order. We may think of this as root extraction in a factor group: when the adversary algorithm gets an input element  $Y$ , this represents an element  $\bar{Y}$  in the quotient group  $G/U$ , and the adversary’s task actually is to extract a non-trivial ( $e$ ’th) root of  $\bar{Y}$  in  $G/U$ . He must, however, demonstrate that his answer when raised to the  $e$ ’th

power represents the same element as does  $Y$ . We require he does this by also producing  $\mu$ .

If we specialize to the RSA case, i.e.,  $G = (\mathbb{Z}/n\mathbb{Z})^\times$  for an RSA modulus  $n$ , it may seem that the root assumption as we defined it here would make an even stronger requirement than the standard strong RSA assumption [2,14] – since the adversary in our case is given  $l_G$  and does not have to find a root of  $Y$ , but of  $\mu Y$  for any  $\mu \in U$ . This is not the case, however. We now show that RSA moduli can be constructed such that our assumptions are satisfied for these groups, based only on the strong RSA assumption in its standard form. Suppose we make a  $k$ -bit modulus  $n = pq$  such that  $\gcd(p-1, q-1) = 2$ . We choose  $C(k)$  as some super-polynomial function much less than  $2^k$ , for instance  $C(k) = 2^{k/10}$ , and we set  $l(k) = k$ . We construct  $p, q$  such that the factor of  $(p-1)(q-1)$  with prime factors less than  $C(k)$  is in  $O(k)$  (this factor is  $l_G$ , where  $G = (\mathbb{Z}/n\mathbb{Z})^\times$ ). We then set  $G = (\mathbb{Z}/n\mathbb{Z})^\times$  and  $\text{descr}(G) = \{n, l_G\}$ . Now, the root assumption (in its asymptotic form) turns out to be equivalent to the standard strong RSA assumption. First note that it makes little difference whether  $l_G$  is known since it can be guessed with significant probability. Then suppose algorithm  $A$  on input  $Y, n, l_G$  finds  $X, e, \mu$  such that  $Y = \mu X^e, \mu^{l_G} = 1$ . Now, if there is non-negligible probability that  $\mu \neq \pm 1$ , we can use  $\mu, l_G$  to factor  $n$ , namely we first factor  $l_G$  and then we can find an element  $\tilde{\mu}$  of known prime order  $s$ . If  $s = 2$ ,  $\tilde{\mu}$  is a non-trivial square root of 1 and  $\gcd(\tilde{\mu} - 1, n)$  is a factor in  $n$ . But if  $s$  is odd, it cannot divide both  $p-1$  and  $q-1$  and therefore  $\tilde{\mu}$  must be congruent to 1 modulo one of  $p$  or  $q$  and be different from 1 modulo the other. Hence, also in this case,  $\gcd(\tilde{\mu} - 1, n)$  is a non-trivial factor of  $n$ . On the other hand, if  $\mu = \pm 1$  with non-negligible probability, we can solve the strong RSA problem: given input  $h \in (\mathbb{Z}/n\mathbb{Z})^\times$ , we choose a random bit  $b$  and give  $(-1)^b h$  as input to  $A$ . Since  $A$  receives the same input distribution as usual, it outputs a non-trivial root of  $(-1)^b Y$  or  $-(-1)^b Y$  with good probability. Since  $A$ 's choice of the sign cannot be correlated to our choice of  $b$ , we obtain a root of  $Y$  with non-negligible probability.

Note that a special case of this construction of  $n$  is when  $n = pq$  is a safe prime product, i.e.,  $(p-1)/2, (q-1)/2$  are primes, but evidently the construction covers a much larger class of moduli.

### 3 Some Definitions

We will often use the concepts of zero-knowledge and computational/statistical indistinguishability. For definitions of these, refer to [16]. The definitions below are all in the exact security style. It is straightforward to derive asymptotic type definitions from the exact-security style ones.

We then define the type of commitment scheme we will look at. Our commitments will be statistically (unconditionally) hiding and computationally binding. Concretely, a commitment scheme consists of a probabilistic polynomial time *key generator*  $\mathcal{H}$ , which on input  $1^k$  outputs a public key  $pk$  and a witness  $w$ . We let  $L_H$  be the set of public keys that  $\mathcal{H}$  can produce as an output,  $w$  is

an NP-witness to the fact that  $pk \in L_H$ . The scheme also defines an algorithm **commit** that takes as inputs  $pk$ , a string  $s$  to be committed to, and a random string  $r$ , both of lengths that are fixed polynomials in  $k$ . The output is a commitment to  $s$ ,  $\text{commit}_{pk}(s, r)$  and a string  $u$ . Finally we have an algorithm **verify** that takes inputs  $pk$ , commitment  $c$ , and strings  $s, u$ , where the output (denoted  $\text{verify}_{pk}(c, s, u)$ ) is *accept* or *reject*.

Such a scheme can be used by a committer  $C$  and a receiver  $R$  as follows: in the *set-up* phase,  $R$  runs  $\mathcal{H}$  to get  $pk, w$ , sends  $pk$  and uses  $w$  to give a zero-knowledge interactive proof [16] that  $pk \in L_H$ .  $C$  can commit to  $s$  by running **commit** on  $pk$ ,  $s$  and random input  $r$ , sending  $c = \text{commit}_{pk}(s, r)$  to  $R$  and keeping  $r$  secret. Opening takes place by revealing  $s, r$  to  $R$ , who can then check that  $\text{verify}_{pk}(c, s, r) = \text{accept}$ .

We then require the following:

**Hiding:** For  $pk \in L_H$ , uniform  $r, r'$  and any  $s, s'$ , we have that the distributions of  $\text{commit}_{pk}(s, r)$  and  $\text{commit}_{pk}(s', r')$  are (statistically) indistinguishable (as defined in [16]).

**Binding:** We say that binding is  $(t(k), \epsilon(k))$ -secure if it holds that for any  $C$  running in time at most  $t(k)$ , the probability that  $C$  on input  $pk$  computes  $s, r, s', r'$  such that  $\text{commit}_{pk}(s, r) = \text{commit}_{pk}(s', r')$  and  $s \neq s'$ , is at most  $\epsilon(k)$ .

We will also need to consider proofs of knowledge in the following. For this, we use a modification of the definition of Bellare and Goldreich, in the version for computationally convincing proofs of knowledge [6]. Let a binary relation  $R$  be given, where a prover  $P$  and a verifier  $V$  are both probabilistic polynomial time interactive Turing machines. Intuitively, the prover's claim is that for a given common input  $c$ , he knows  $w$  such that  $(c, w) \in R$ .

To define this in our setting, we cannot use the original definition in [6] without change. This is because it asks that the soundness of the protocol holds for *all* (large enough) instances  $c$ . In our scheme, this is more than we can reasonably ask for: in our case, one may think of  $c$  as a commitment and  $w$  as the string  $P$  can use to open  $c$ . Furthermore, the scenario is that  $P$  sees the public key of the scheme, produces the commitment and then tries to prove he knows how to open it. But this proof is only computationally convincing in our case, so a cheating prover may have some chance of producing, based on the public key, a commitment he cannot open, but where the proof nevertheless is successful with large probability. This can typically happen if the prover manages to compute some trapdoor information associated with the public key. This information can always be guessed with non-zero probability and so the problem cannot be completely avoided, but we can at least require that it occurs with only small probability. In our definition of soundness, therefore, we first let the prover see a public piece of information, he then produces  $c$  and conducts the proof. A cheating prover  $P^*$  wins if the standard soundness requirement fails for this  $c$ , and we are satisfied with the proof system if within some time bound  $P^*$  can only win with some bounded (small) probability.



For the above definition we need to consider a *relation generator*, algorithm  $\mathcal{R}$ , that takes  $1^k$  as an input and produces as an output a description of a binary relation  $R$ . By this we mean a string containing information is sufficient to sample efficiently random pairs  $(c, w) \in R$  and to test membership in  $R$  efficiently. We will use  $R$  to denote both this description and the relation itself. For instance,  $\mathcal{R}$  might be the key generator for a commitment scheme, and we can think of the public key  $pk$  as defining a relation consisting of pairs  $(c, (s, r))$  for which  $\text{verify}_{pk}(c, s, r) = \text{accept}$ .

A prover in our setting is a machine  $P$  who gets  $R$  as an input, outputs a string  $c$  and finally conducts the interactive proof with a verifier  $V$  using  $R, c$  as common input. For convenience, we want to be able to refer to  $P$ 's strategy when executing the proof as a separate machine. Therefore, from  $P$  we define a machine  $P_{\text{view}}$  which starts in the state  $P$  is in after having seen view  $\text{view}$  and having produced  $c$ .  $P_{\text{view}}$  then conducts the protocol with  $V$  following  $P$ 's algorithm. The view  $\text{view}$  contains all inputs, messages exchanged and random coins so in particular  $c$  is determined by  $\text{view}$ . Note that the distribution of  $\text{view}$  is taken over the random coins of both  $P$  and  $\mathcal{R}$ . We let  $\epsilon_{\text{view}, P}$  be the probability with which  $P_{\text{view}}$  makes  $V$  accept, i.e.  $\epsilon_{\text{view}, P}$  is  $P$ 's probability to make  $V$  accept, conditioned on  $\text{view}$ .

An *extractor* will be a machine  $M$  that gets  $R, c$  as an input, has black-box access to  $P_{\text{view}}$  for some  $\text{view}$  consistent with  $c$ . and computes a witness  $w$  such that  $(c, w) \in R$ . The intuition is that the prover “knows”  $w$  if we can use  $M$  to extract it from him. To measure this, we need a *knowledge error* function  $\kappa()$ . Intuitively,  $\kappa(k)$  is the probability that the prover can cheat on input generated from security parameter value  $k$ , i.e., make  $V$  accept while knowing nothing about  $w$ .

**Definition 1.** For some given cheating prover  $P^*$ , extractor  $M$  and polynomial  $p()$ , we say  $M$  fails on view  $\text{view}$  if  $\epsilon_{\text{view}, P^*} > \kappa(k)$ , if the expected running time of  $M$  using  $P_{\text{view}}^*$  as oracle, is greater than  $\frac{p(k)}{\epsilon_{\text{view}, P^*} - \kappa(k)}$ .

This definition is motivated by the fact that the standard knowledge soundness requirement from [6] says that the extractor must run in expected time *at most* the bound in the definition. Note that in any situation where  $P^*$  has produced  $c$  having seen  $\text{view}$ , it is well defined whether  $M$  fails or not. Intuitively, one may think of this as saying that if  $M$  does not fail in a given situation, then  $P^*$  really “must know” a witness for  $c$ , in order to make  $V$  accept with a probability better than  $\kappa(k)$ .

**Definition 2.** Let  $\mathcal{R}$  be a probabilistic polynomial time relation generator, and let a protocol  $(P, V)$ , a knowledge extractor  $M$ , polynomial  $p()$  and knowledge error function  $\kappa()$  be given. Consider the following experiment with input  $k$ :  $R := \mathcal{R}(1^k), c := P^*(R)$  (this defines view  $\text{view}$ ). We define the advantage of  $P^*$ ,  $\text{Adv}_{\kappa, M, p}(P^*, k)$  as the probability that  $M$  fails on the view generated by this experiment. This probability is taken over the random coins of  $R, P^*$ .

Finally, for a relation  $R$ , we let, as usual,  $L_R = \{c \mid \exists w : (c, w) \in R\}$ . We are now ready to define computationally convincing proofs of knowledge:



**Definition 3.** Let  $\mathcal{R}$  be a probabilistic polynomial time relation generator. We say that  $(P, V)$  is a computationally convincing proof of knowledge for  $\mathcal{R}$ , with knowledge error  $\kappa()$ , failure probability  $\nu()$  and time bound  $t()$ , if the following hold:

**Knowledge Completeness.** The honest prover  $P$  receives  $R \leftarrow \mathcal{R}(1^k)$ , produces  $(c, w) \in R$ , sends  $c$  to  $V$  and finally conducts the protocol with  $V$ , who accepts with overwhelming probability in  $k$ .

**Knowledge Soundness.** There exists a polynomial  $p()$  and an extractor  $M$ , such that for all provers  $P^*$  running in time at most  $t(k)$ ,  $\text{Adv}_{\kappa, M, p}(P^*, k) \leq \nu(k)$ .

## 4 The Commitment Scheme

Based on the above model, the goal is to make a commitment scheme with protocols to verify various claims on committed values. The basic scheme is that the verifier  $V$  (the receiver of commitments) will run  $\mathcal{G}$  and send  $\text{descr}(G)$  (and more information to be described later) to the prover  $P$  (the committer).

**Set-Up.**  $V$  runs  $\mathcal{G}(1^k)$  and chooses a random element  $h \in G$ , such that  $\text{ord}(h)$  is  $C(k)$ -rough (this can be done by raising a random element to power  $l_G$ ). Now  $V$  sets  $g = h^\alpha$ , where  $\alpha$  is randomly chosen in  $[0..2^{2B+k}]$ .  $V$  sends  $\text{descr}(G), g, h$  to  $P$  and proves that  $g \in \langle h \rangle$ , by the standard zero-knowledge discrete log protocol with binary challenges: in one iteration of this,  $V$  sends  $a = h^R$  for a random  $R \in [0..2^{2B+2k}]$ .  $P$  selects a random bit  $b$ , and  $V$  replies with  $z = R + b\alpha$ .  $P$  checks that  $h^z = ag^b$ . Repeating this  $k$  times results in a soundness error of  $2^{-k}$ , and the protocol is easily seen to be statistical zero-knowledge. This is not a very efficient solution, but it only needs to be done once and only in the set-up phase.

**Commit.** To commit to an integer  $x$ ,  $P$  chooses  $r$  at random in  $[0..2^{B+k}]$ , sends  $c = g^x h^r$  to  $V$ , and stores  $x, r$  for later use.

**Open.** To open a commitment,  $P$  must send  $x, r, \mu$  such that  $c = \mu g^x h^r$  and  $\mu^{l_G} = 1$ . An honest prover can always use  $\mu = 1$ . Although this gives a dishonest prover extra freedom, this in no way makes the commitment scheme weaker: the binding property still holds, as we argue below. Indeed, recalling our comments on the root assumption, one may think of the scheme as taking place in the quotient group  $G/U$  where  $U$  is the subgroup in  $G$  consisting of all elements of  $C$ -smooth order. From this point of view, the opening condition simply ensures that the prover opens something representing the same element as  $c$  in  $G/U$  (The quotient group is defined canonically so that  $c \equiv \bar{c} \pmod{U}$  iff there is a  $\mu \in U$  such that  $\bar{c} = \mu c \in G$ ).

As for hiding, note that  $P$  verifies initially that  $g \in \langle h \rangle$ . Hence, since  $r$  is chosen with bit length at least  $k + \log_2(\text{ord}(h))$ ,  $c$  is statistically close to uniform in  $\langle h \rangle$ , for any value of  $x$ .

As for binding, we consider any prover  $P^*$  who can create  $c$  and the corresponding valid distinct openings,  $(\mu, x, r)$  and  $(\mu', x', r')$ . It follows that we get

$\mu g^x h^r = c = \mu' g^{x'} h^{r'}$ . Recall that  $V$  creates  $g$  as  $g = h^\alpha$ . Plugging this in and raising both sides of the equation to  $l_G$ , we get that  $h^{l_G(\alpha(x-x')+(r-r'))} = 1$ . We can write  $\alpha = q \cdot \text{ord}(h) + \text{res}$  for integers  $q, \text{res}$  with  $0 \leq \text{res} < \text{ord}(h)$ . Then from  $P^*$ 's point of view,  $\text{res}$  is uniquely determined from  $g$ , whereas there is an exponentially small amount of information on  $q$  (the only source of information is the proof that  $g \in \langle h \rangle$  which is statistical zero-knowledge). So  $P^*$ 's choice of  $x, x', r, r'$  is (almost) independent of  $q$ . It follows  $l(\alpha(x-x')+(r-r')) = 0$  (as an integer) with probability exponentially small in  $k$ . Assuming this number is indeed non-zero,  $\text{ord}(h)$  must divide  $M := \alpha(x-x')+(r-r')$  (since the order is  $C(k)$ -rough).

We can now use  $P^*$  to break the root assumption as follows: given input  $\text{descr}(G), h \in G$ , we choose  $g$  as  $V$  would have done it, send  $\text{descr}(G), h, g$  to  $P^*$  and execute in the normal way the proof that  $g \in \langle h \rangle$ . With probability of at least  $1/l(k)$ ,  $h$  will have  $C(k)$ -rough order and everything has the same distribution as in a normal execution of the commitment scheme. Given that  $P^*$  breaks the binding property as described above, this allows us (except with negligible probability) to compute  $M$ , a multiple of the order of  $h$ . Now choose any  $t$  that is relatively prime to  $M$  and output  $h^{t^{-1} \bmod M}$  and  $t$ .

Summarizing, we have:

**Theorem 1.** *Under the root assumption, the above scheme is an unconditionally hiding and computationally binding commitment scheme. If the root assumption is  $(t(k), \epsilon(k))$ -secure, then the binding property is  $\left(t(k), \frac{\epsilon(k)l(k)}{1-2^{-\gamma k}}\right)$ -secure for some constant  $\gamma$ .*

## 5 Associated Protocols

### 5.1 Proving You Know How to Open

The following protocol can be used by  $P$  to show that he can open a given commitment  $c = g^x h^r$ .

We will assume that  $x$  is in  $[-T..T]$  where  $T(> 0)$  is a public constant.  $T$  can be chosen arbitrarily large, and is only used to control the size of the prover's random choices, this allows an honest prover to ensure that the protocol hides the value of  $x$ , whenever  $-T \leq x \leq T$ . In any application of the scheme, one simply chooses  $T$  large enough to accommodate any choice of  $x$  an honest prover would need to make in the given scenario. The protocol guarantees an honest verifier that  $-TC(k)(2^k + 2) \leq x \leq TC(k)(2^k + 2)$ . To prove  $x$  is in some other (smaller) interval, other techniques exist, see e.g. [4].

1.  $P$  chooses  $y \in [0..TC(k)2^k]$ ,  $s \in [0..C(k)2^{B+2k}]$  at random and sends  $d = g^y h^s$  to  $V$ .
2.  $V$  chooses at random  $e \in [0..C(k)]$  and sends to  $P$ .
3.  $P$  sends  $u = y + ex, v = s + er \in \mathbb{Z}$ .  $V$  checks that  $g^u h^v = dc^e$  and that  $[-TC(k)..TC(k)(2^k + 1)]$ .

Define a relation generator  $\mathcal{R}$  as follows: run  $\mathcal{G}(1^k)$  to get  $G$ , choose  $h \in H$  with  $C(k)$ -rough order, set  $g = h^\alpha$  and output  $\text{descr}(G), g, h$ . Then define the relation  $R = \{(c, (\mu, x, r)) \mid c, b \in G, c = \mu g^x h^r, \mu^{l_G} = 1, x \in [-TC(k)(2^k + 2)..TC(k)(2^k + 2)]\}$ .

We now analyze to what extent the protocol above satisfies our definition of knowledge soundness, in particular for which knowledge error functions is the definition satisfied. Accordingly, let  $\kappa()$  be any knowledge error function, such that  $\kappa(k) \geq 4/C(k)$  for all  $k$ . We then must define an extractor  $M$ . Let a polynomial time prover  $P^*$  be given and let  $\text{view}$  be any view  $P^*$  may have after having produced a commitment  $c$ . Now, it can be shown that since there are  $C(k)$  different challenges, then if  $\epsilon_{\text{view}, P^*} > \kappa(k) \geq 4/C(k)$ , standard rewinding techniques allow us to obtain in expected polynomial time a situation where, for a given  $d$ ,  $P^*$  has correctly answered two different values  $e$  and  $e'$  with numbers  $u, v$  and  $u', v'$ , so we get  $g^{u-u'} h^{v-v'} = c^{e-e'}$ . Let  $\text{Rewind}$  be a (probabilistic) procedure that creates  $e, e', u, v, u', v'$  in this way. A concrete algorithm for  $\text{Rewind}$  is given in Appendix A. It runs in expected time  $56/\epsilon_{\text{view}, P^*}$ , counting the time to do the protocol once with  $P^*$  as one step<sup>1</sup>.

Assume without loss of generality that  $e > e'$  and suppose that  $(e - e')$  divides both  $(u - u')$  and  $(v - v')$ . We now see that the element  $\mu = g^{\frac{u-u'}{e-e'}} h^{\frac{v-v'}{e-e'}} c^{-1}$  satisfies that  $\mu^{e-e'} = 1$ . Since  $e - e' < C(k)$ , it follows that  $\text{ord}(\mu)$  is  $C(k)$ -smooth so that  $\mu^{l_G} = 1$ . So  $c$  can be correctly opened by sending  $(u - u')/(e - e'), (v - v')/(e - e'), \mu$ . Moreover,  $V$ 's check on the size of  $u, u'$  implies that  $(u - u')/(e - e')$  is in the required interval. A set of values  $e, e', u, u', v, v'$  is said to be *bad* if  $e - e'$  does not divide both  $u - u'$  and  $v - v'$ . The extractor  $M$  simply repeats calling  $\text{Rewind}$  (for this same  $c$ ) until it gets a set of good values. We will analyze knowledge soundness with this  $M$  and the polynomial  $p(k)$  from the definition set to the constant of 112. We start with a lemma that gives an exact bound on the security.

**Lemma 1.** *Let  $\mathcal{R}, (P, V), \kappa, M$  and  $p()$  be as defined above. Given any prover  $P^*$ , there exists an algorithm  $A(P^*)$  that solves the root problem defined by  $\mathcal{G}(1^k)$  with probability  $\frac{\text{Adv}_{\kappa, M, P}(P^*, k)}{9l(k)}$  if  $k \geq 6$ , and runs in time  $448 \cdot t_{P^*}(k)/\kappa(k)$  where  $t_{P^*}(k)$  denotes the running time of  $P^*$  ( $1/l(k)$  is an lower bound on the probability that a random element in  $G$  has  $C(k)$ -rough order ).*

*Proof.* The algorithm claimed does the following: receive  $G, h$  as an input. Set  $g = h^\alpha$  for random  $\alpha \in [0..2^{2B+k}]$ . We send  $g, h$  to the adversary, call  $\text{Rewind}$  and hope that we get a set of bad values. However, we will only allow  $\text{Rewind}$  to do the protocol with the prover at most  $448/\kappa(k)$  times. If  $\text{Rewind}$  runs longer than this, we abort it and stop. If we obtained a set of bad values, we attempt to compute a root of  $h$  as described below.

<sup>1</sup> Note that this is not completely trivial, as  $P^*$  is probabilistic: although its average success probability is  $\epsilon_{\text{view}, P^*}$ , it may not be equally successful for all choices of random coins. It is essential to get the claimed expected time that  $\epsilon_{\text{view}, P^*} > 4/C(k)$ , and not just  $> 1/C(k)$

It is immediately clear that this algorithm has the claimed running time. We now look at the success probability. We will assume that  $h$  has  $C(k)$ -rough order. Since this happens with probability of at least  $1/l(k)$ , it is enough to show that the success probability under this assumption is at least the bound claimed times  $l(k)$ .

Note that the distribution of  $G, h, g$  that  $P^*$  receives here is exactly the same as in the real commitment scheme. Hence the probability of producing a view for which  $M$  fails, is exactly  $\text{Adv}_{\kappa, M, p}(P^*, k)$ . Note also that given any view  $\text{view}$  where  $M$  fails, it must be the case that the values produced by **Rewind** are bad with probability of at least  $1/2$ . If this was not the case, then  $M$  could expect to find a way to open  $c$  after calling **Rewind** twice, which takes expected time  $112/\epsilon_{\text{view}, P^*} \leq p(k)/(\epsilon_{\text{view}, P^*} - \kappa(k))$  so this would contradict the fact that  $M$  fails on view. So let  $E$  be the event that  $M$  fails on view and **Rewind** has returned a set of bad values. We now make the following

**Claim:** Given that  $E$  occurs, we can solve the root problem with probability of at least  $1/2 - 2^{-k}$ .

To see this, recall that **Rewind** returns  $e, e', u, u', v, v'$  such that  $g^{u-u'}h^{v-v'} = c^{e-e'}$ , and we have that  $e - e'$  does not divide both  $u - u'$  and  $v - v'$ . If we plug in that  $g = h^\alpha$ , we get

$$h^{\alpha(u-u')+(v-v')} = c^{e-e'}$$

We then split in two cases:

**Case 1:  $e - e'$  does not divide  $\alpha(u - u') + (v - v')$ .**

In this case, let  $\beta = \gcd(e - e', \alpha(u - u') + (v - v'))$  (where by assumption  $\beta < e - e' \leq C(k)$ ). Choose  $\gamma, \delta$  such that

$$\gamma(e - e') + \delta(\alpha(u - u') + (v - v')) = \beta$$

We then get that

$$h^\beta = h^{\gamma(e-e')+\delta(\alpha(u-u')+(v-v'))} = (h^\gamma c^\delta)^{e-e'}.$$

If we set  $\tilde{\mu} = (h^\gamma c^\delta)^{(e-e')/\beta} h^{-1}$ , it is clear that  $\tilde{\mu}^\beta = 1$ , so since  $\beta < C(k)$ ,  $\text{ord}(\tilde{\mu})$  is  $C(k)$ -smooth so that  $\tilde{\mu}^{l^G} = 1$ . Furthermore

$$h\tilde{\mu} = (h^\gamma c^\delta)^{(e-e')/\beta}$$

So in this case, we may output  $h^\gamma c^\delta, (e - e')/\beta, \tilde{\mu}$ , which is a solution to the root problem as we defined it earlier.

**Case 2:  $e - e'$  divides  $\alpha(u - u') + (v - v')$ .**

Note that even in this case, we still have that  $e - e'$  does not divide both  $u - u'$  and  $v - v'$ . The goal will be to show that since the adversary does not know full information about our choice of  $\alpha$ , this case happens with probability at most  $(1/2 - 2^{-k})$ , given that  $E$  occurs. Hence the previous case where we could solve the root problem happens with large probability, given  $E$ . Let  $q$  be some prime factor in  $e - e'$  such that  $q^j$  is the maximal  $q$ -power dividing  $e - e'$ , and at least one of  $u - u', v - v'$  are non-zero modulo  $q^j$  (such a  $q$

must exist since  $e - e'$  does not divide both of  $u - u', v - v'$ . Note that if  $q^j$  divides  $u - u'$ , it would have to divide  $v - v'$  as well, which is a contradiction. So  $u - u' \not\equiv 0 \pmod{q^j}$ . We can then write  $\alpha = y + z \cdot \text{ord}(h)$ , where  $y = \alpha \bmod \text{ord}(h)$ . Note that  $g$  represents all information the adversary has about  $\alpha$  and  $y$  is uniquely determined from  $g$ , whereas  $z$  is completely unknown. Now, if indeed  $q^j$  divides  $\alpha(u - u') + (v - v')$ , we have

$$\alpha(u - u') + (v - v') = z(u - u')\text{ord}(h) + y(u - u') + (v - v') = 0 \pmod{q^j}$$

Note that since  $q < C(k)$  we have  $\text{ord}(h) \not\equiv 0 \pmod{q}$ . Now, from the adversary's point of view,  $z$  is chosen uniformly among at least  $2^{B+k}$  values, and must satisfy the above equation in order for the bad case to occur. The number of solutions modulo  $q^j$  of this equation is at most  $\gcd((u - u')\text{ord}(h), q^j)$ . This number is a power of  $q$ , but is at most  $q^{j-1}$ . Then, since  $2^{B+k}$  is larger than  $q^j$  by a factor of at least  $2^k$ , it follows that the distribution of  $z \bmod q^j$  is statistically close to uniform in  $\mathbb{Z}/q^j\mathbb{Z}$ . In fact, the probability that  $z$  satisfies the equation is at most  $1/q - 2^{-k} \leq 1/2 - 2^{-k}$ . The claim above now follows.

Summarizing, we therefore have that for every view  $\text{view}$  where  $M$  fails, running Rewind will fail to solve the root problem with probability at most  $1 - (1/2 - 2^{-k})/2 = 3/4 + 2^{-k-1}$ . The expected number of executions of  $P^*$  needed to run rewind is at most  $56/\epsilon_{\text{view}, P^*} \leq 56/\kappa(k)$ . Thus Rewind is allowed to run for at least 8 times its expected running time, and so by the Markov rule it will run for longer with probability at most  $1/8$ . Since the probability that  $\text{view}$  is bad in the first place is  $\text{Adv}_{\kappa, M, p}(P^*, k)$ , the success probability of  $A(P^*)$  is  $\text{Adv}_{\kappa, M, p}(P^*, k)(1 - 1/8 - 3/4 - 2^{-k-1}) \geq \text{Adv}_{\kappa, M, p}(P^*, k)/9$  if  $k \geq 6$ . This finishes the proof.

Next we have:

**Theorem 2.** *If the root assumption is  $(t'(k), \epsilon(k))$ -secure, the above protocol is a computationally convincing proof of knowledge for  $\mathcal{R}$  with knowledge error  $\kappa(k)$ , time bound  $t(k)$  and failure probability  $\nu(k)$ , where  $\nu(k) = 9\epsilon(k)l(k)$ ,  $t(k) < t'(k)/448$  and  $\kappa(k) = \max(4/C(k), 448t(k)/t'(k))$ . If  $-T \leq x \leq T$  (as it will be when the prover is honest), the protocol is honest verifier statistical zero-knowledge.*

*Remark 1.* There are a number of known techniques by which a protocol that is zero-knowledge in general can be constructed from an honest verifier zero-knowledge protocol.

*Remark 2.* Note that a prover playing against the commitment scheme as defined above will see both the public key  $pk$  and a zero-knowledge proof from  $V$  that  $pk$  was correctly chosen, whereas a prover in the proof of knowledge definition only sees the public key. This makes no difference, however, since the proof is statistical zero-knowledge and could always be simulated.

*Proof.* Completeness of this protocol is clear. It is honest verifier statistical zero-knowledge: to simulate we can choose at random  $u \in [0..TC(k)2^k]$ ,  $v \in [0..C(k)2^{B+2k}]$ ,  $e \in [0..C(k)]$  and set  $d = g^u h^v$ . The rest follows immediately from the preceding lemma.

## 5.2 A Multiplication Protocol

Using techniques similar to those above, we can also get a protocol for proving that three given commitments  $c_1, c_2, c_3$  contain numbers  $x_1, x_2, x_3$  such that  $x_3 = x_1 x_2$ . We assume that  $c_i = g^{x_i} h^{r_i}$ , and as before that the  $x_i$ 's are numerically smaller than  $T$ . Note that then we have  $c_3 = c_1^{x_2} h^{r_3 - x_2 r_1}$ , i.e., using  $c_1$  as the “base element” for the commitment  $c_3$ , it will contain the same value as does  $c_2$  using  $g$  as base. So if the prover can convince us of this and also that he can open  $c_1$ , it will follow that  $x_3 = x_1 x_2$ . This is the idea behind the protocol below:

1.  $P$  chooses at random  $y_1, y \in [0..C(k)T2^k]$ ,  $s_1, s_2 \in [0..C(k)2^{B+2k}]$ ,  $s_3 \in [0..C(k)T2^{B+2k}]$  and sends  $d_1 = g^{y_1} h^{s_1}$ ,  $d_2 = g^y h^{s_2}$ ,  $d_3 = c_1^y h^{s_3}$  to  $V$ .
2.  $V$  chooses at random  $e$  between 0 and  $C(k)$  and sends to  $P$ .
3.  $P$  sends  $u_1 = y_1 + ex_1$ ,  $u = y + ex_2$ ,  $v_1 = s_1 + er_1$ ,  $v_2 = s_2 + er_2$  and  $v_3 = s_3 + e(r_3 - x_2 r_1)$ .  $V$  checks that  $g^{u_1} h^{v_1} = d_1 c_1^e$ ,  $g^u h^{v_2} = d_2 c_2^e$ , and  $c_1^u h^{v_3} = d_3 c_3^e$ .

Define a relation generator  $\mathcal{R}_{\text{mult}}$  as follows: run  $\mathcal{G}(1^k)$  to get  $G$ , choose  $h \in G$  with  $C(k)$ -rough order, set  $g = h^\alpha$  and output  $\text{descr}(G), g, h$ . Then define the relation  $R_{\text{mult}} = \{((c_1, c_2, c_3), (x_1, r_1, b_1, x_2, r_2, b_2, x_3, r_3, b_3)) \mid c_i, b_i \in G, c_i = \mu_i g^{x_i} h^{r_i}, \mu_i^l = 1, i = 1, 2, 3\}$ . This leads to:

**Theorem 3.** *If the root assumption is  $(t'(k), \epsilon(k))$ -secure, the above protocol is a computationally convincing proof of knowledge for  $\mathcal{R}$  with knowledge error  $\kappa(k)$ , time bound  $t(k)$  and failure probability  $\nu(k)$ , where  $\nu(k) = 9\epsilon(k)l(k)$ ,  $t(k) < t'(k)/448$  and  $\kappa(k) = \max(4/C(k), 448t(k)/t'(k))$ . If  $-T \leq x_1, x_2, x_3 \leq T$  (as they will be when the prover is honest), the protocol is honest verifier statistical zero-knowledge.*

For the space limitation, we omit the proof, which can be easily derived from the proof of Theorem 2.

## 6 What Is the Major Difference from the Earlier Proof in [14]?

For completeness, we briefly indicate here what is mainly different from the earlier work [14] in terms of the proof of soundness. As mentioned above, the main gap we fill here does not appear in the proofs in related works [8, 3, 10, 4]. This is because the gap only appears in the proofs for protocols associated with commitment using plural bases (i.e.,  $c = g^s h^r$  such as in [14]).

These protocols suggested in [14] are very similar to the ones we suggest here, in particular they have the same 3-move form, with a challenge  $e$  from the verifier

as the second message. So [14] uses a rewinding argument as we do here, to obtain correct answers from the prover to challenges  $e, e'$ . However, a problem occurs in the last part of the proof, which corresponds to the last case in our analysis, that is, Case 2: “ $(e - e')$  divides  $\alpha(u - u') + (v - v')$ ”. Translated into our notation, we have now  $(e - e')$ ,  $u - u'$ , and  $v - v'$  such that  $h^{\alpha(u - u') + (v - v')} = c^{e - e'}$ . If  $e - e'$  divides both of  $(u - u')$  and  $(v - v')$ , we are essentially done since we then have  $c = \mu g^{(u - u')/(e - e')} h^{(v - v')/(e - e')}$  where  $\mu^{e - e'} = 1$ . In the earlier work [14], it is claimed that the adversary can make Case 2 occur only with negligible probability unless  $e - e'$  divides both of  $(u - u')$  and  $(v - v')$ , because he doesn't have enough information about  $\alpha$ , where  $g = h^\alpha$ . However, if  $e - e'$  is a small number, then this case may in fact happen with significant probability, even without  $e - e'$  dividing both  $u - u'$  and  $v - v'$ . This problem was not taken into account in [14]. Later in the full paper version of [14], it was shown that when the knowledge extractor rewinds  $P^*$  and makes him output  $e, e'$ , it only happens with negligible probability that  $e - e'$  is *small*, see [15]. However, even if  $e - e'$  is large, there is still a problem: if  $e - e'$  has a small prime factor  $p$ , there may be a significant probability that  $p$  divides  $\alpha(u - u') + (v - v')$  whereas it does not divide both  $u - u'$  and  $v - v'$ . The additional idea we provide here essentially fills this gap, and indeed seems necessary for this type of proof to go through.

As for [8, 3, 10, 4], what corresponds to Case 2 is the event “ $(e - e')$  divides  $(u - u')$ ”; there is no gap to go to  $c = \mu g^{(u - u')/(e - e')}$ , where  $\mu^{e - e'} = 1$ . Hence, the related works above do not suffer from the problem we need to consider.

## 7 Applying the Scheme in Class Groups and Beyond

We do not give any detailed introduction to class groups here – or more precisely, class groups of quadratic number fields. It is enough to know, that each such group is defined by a single number, the *discriminant*  $\Delta$ . Given this number, one can choose elements in the group and compute the group and inversion operations. Finding the order of the class group (the class number) from  $\Delta$  appears to be a hard problem, and is at least as hard as factoring  $\Delta$  (if  $\Delta$  is composite). Therefore, root extraction also appears to be a hard problem, and it seems reasonable to conjecture that if  $\Delta$  is chosen randomly from a large set of values, then the class number will contain large and random prime factors, and will not have a very large factor consisting of only small primes. Various heuristics (but no proofs) supporting this are known. All of this together makes it a reasonable conjecture that class groups constructed from large, random discriminants would satisfy the assumptions we made in the beginning, for some appropriate choice of  $C(k)$ <sup>2</sup>.

There is one difficulty, however: we have assumed that  $G$  can be generated such that  $l_G$ , the order of the subgroup  $U$  of  $C(k)$ -smooth elements is known. Unfortunately, there is no known way to do this for class groups.

<sup>2</sup> There are some heuristics known that describe how the factorization of a class number can be expected to behave,  $C(k)$  should be chosen with this in mind.

One way to solve this is to observe that we have only used  $l_G$  in order to verify membership in  $U$ . So we can do the following: assume we can choose  $C(k)$  and  $l(k)$  such that  $C(k) > l(k)$  and (as usual) such that  $l(k)$  is a polynomial and the order of  $U$  is less than  $l(k)$ . Now we replace  $l_G$  in all descriptions and proofs by  $l(k)!$ . This works because  $l(k)!$  is guaranteed to be a multiple of  $l_G$  and all its prime factors are at most  $l(k)$ , and so are less than  $C(k)$ .

Another possibility is to rely on an additional intractability assumption, namely that given  $\text{descr}(G)$ , it is hard to find a non-trivial element in  $U$ . This seems to be a reasonable assumption in many settings: indeed  $U$  is an extremely small subgroup, so a random choice will succeed with negligible probability. Moreover, in the case of class groups with a composite discriminant, finding an element of order 2 is equivalent to factoring the discriminant. With this assumption, all the cases where we needed to know  $l_G$  occur with negligible probability, and can be ignored.

## References

1. F. Bao: *An efficient verifiable encryption scheme for encryption of discrete logarithm*, In CARDIS'98, LNCS 1820, pp.213–220, 2000.
2. N. Baric and B. Pfitzmann: *Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees*, In EUROCRYPT'97, LNCS 1233, pp.480–494, 1997.
3. F. Boudot and J. Traoré: *Efficient publicly verifiable secret sharing schemes with fast or delayed recovery*. In 2nd ICICS, LNCS 1726, pp.87–102. 1999.
4. F. Boudot: *Efficient Proof that a Committed Number Lies in an Interval*, In Eurocrypt LNCS 1807, Springer, 2000.
5. Boudot: presentation at the rump session of Eurocrypt 2000.
6. M. Bellare and O. Goldreich: *Defining proofs of knowledge*, In Crypto 92.
7. R. Cramer and I. Damgård: *Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge be for Free?*, In Crypto 98, LNCS 1462, 1998.
8. A. Chan, Y. Frankel and Y. Tsiounis: *Easy Come - Easy Go Divisible Cash*, In EUROCRYPT'98, pp.561–575 LNCS 1403, 1998.
9. J. Camenisch and M. Michels: *Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes*, In Eurocrypt'99 pp.107–122 LNCS 1592, 1999.
10. J. Camenisch and M. Michels: *Separability and Efficiency for Generic Group Signature Schemes*, In CRYPTO'99 pp.413–430, LNCS 1666, 1999.
11. J. Camenisch and M. Michels: *Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes*, Tech. Report RS-98-29, BRICS, 1999.
12. I. Damgård: *Practical and Provably Secure release of a Secret and Exchange of Signatures*, J.Cryptology, vol. 8, pp.201–222, 1995.
13. E. Fujisaki: *A simple Approach to Secretly Sharing a Factoring Witness in a Publicly-Verifiable Manner*, IEICE Trans. Fund., E85-A, vol.5, May 2002.
14. E. Fujisaki and T. Okamoto: *Statistical Zero-Knowledge Protocols to prove Modular Polynomial Relations*, In Crypto 97, LNCS 1294, 1997.
15. E. Fujisaki and T. Okamoto: *Statistical Zero-Knowledge Protocols to Prove Modular Polynomial Relations*, in IEICE Trans. Fund., E82-A, vol.1 pp. 81–92, Jan. 1999.
16. Goldwasser, Micali and Rackoff: *The knowledge complexity of interactive proof systems*, SIAM J.Computing, vol. 18, pp.186–208, 1989.



17. T. Pedersen: *Non-Interactive and Information Theoretic Secure Verifiable Secret Sharing*, In Crypto 91, LNCS 576, pp. 129–140.
18. P. Paillier: *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, In Eurocrypt'99, LNCS 1592, pp. 223–238, 1999.
19. T. Okamoto and S. Uchiyama: *A New Public-Key Cryptosystem as Secure as Factoring* In Eurocrypt 98, LNCS 1403, 1998.

## A The Rewind Procedure

We are given a prover  $P^*$  who sends a message  $d$ , receives a challenge chosen randomly among  $C$  possibilities, and returns an answer  $z$  that may or may not be correct. We are given that the probability of a correct answer taken over  $P^*$  coins and the choice of  $e$  is at least  $\epsilon > 4/C$ . We want to find correct answers to two different  $e$ -values for a given  $d$  as efficiently as possible.

Of course, the idea is to run the prover, and use rewinding to try to make him answer two different challenges correctly. But to run him, we need to supply random coins. Although we know that the average success probability is  $\epsilon$ , we do not know that  $P^*$  is equally successful with any random input. To get a better view of this, let  $H$  be a matrix with a row for each possible set of random coins for  $P^*$ , and one column for each possible challenge value. Write 1 in an entry if  $P^*$  answers correctly with the corresponding random choices and challenge, and 0 otherwise. Using  $P^*$  as black-box, we can probe any entry we want in  $H$ , and our goal can be rephrased to: find two 1's in the same row. What we know is that the  $\epsilon$  equals the fraction of 1-entries in  $H$ .

It is now apparent that we cannot just search for a 1-entry and then keep looking for another 1 in the same row: if we stumbled across the only 1 in that row, we will never finish. Consider instead the following algorithm *Alg*:

1. Probe random entries in  $H$  until a 1 is found.
2. Then start the following two processes in parallel, and stop when either one stops:
  - $Pr_1$ . Probe random entries in the row in which we found a 1 before, until another 1-entry is found.
  - $Pr_2$ . Repeatedly flip a coin that comes out heads with probability  $\epsilon/w$ , for some constant integer  $w$  (we show how to choose  $w$  later), until you get heads. This can be done by probing a random entry in  $H$  and choosing a random number among  $1, 2, \dots, w$  - you output heads if the entry was a 1 and the number was 1.

This algorithm runs in expected time at most  $w/\epsilon$ , recall that we count access to  $P^*$  as one step. Define a row to be *heavy* if it contains a fraction of at least  $\epsilon/2$  1's. By a simple counting argument, you can see that at least half of the 1's are located in heavy rows. Given that  $Pr_1$  runs in a heavy row, the probability that a probe will succeed is at least  $\frac{C\epsilon/2-1}{C}$  so the expected number of probes it makes is  $T(\epsilon) = C/(C\epsilon/2 - 1)$ . If  $\epsilon \geq 4/C$ , then  $T(\epsilon) \leq 2/\epsilon$ . Moreover, the probability that  $Pr_1$  runs for more time than  $2T(\epsilon)$  is at most  $1/2$ . Assume we choose  $w$  large enough, so that  $Pr_2$  finishes later than  $2T(\epsilon)$  with probability of at least  $1/2$ . It is straightforward to see that  $w = 7$  is sufficient. Then, given

that the row we use is heavy, we have probability of at least  $1/4$  of success, and hence overall probability  $1/8$ .

Therefore, our **Rewind** procedure simply repeats *Alg* until there is success, the expected number of times it will have to do so is 8, and hence the expected total time is  $56/\epsilon$ .

## B Further Extensions

There are many possible variants or extensions with some differences: For example, when  $T_0 < |x|$  is public, we can reduce the computational amount for the protocol by using  $T - T_0$  instead of  $T$ . In the rest of this section, we briefly mention to extensions, though not many due to the space limitation, to non-Abelian groups and verifiable encryptions, where we think the latter might not be so trivial.

For non-Abelian group  $G$ , we can get a similar result if the assumptions are modified as follows: (1) there is a element  $h$  in  $G$  such that its order is  $C$ -rough whereas  $l_G \triangleq [G : \langle h \rangle]$  is  $C$ -smooth, and (2) given  $\text{descr}(G)$  and random  $Y \in G$ , it is difficult to find  $e > 1$ ,  $X \in G$  such that  $Y = \mu_1 X^e \mu_2$  and  $\mu_1^{l_G} = \mu_2^{l_G} = 1$ . Define  $\text{commit}_{pk}(s, r) = g^s h^r$  for  $g \in \langle h \rangle$ , but allow the committer to send  $(s', r')$  when opening commitment  $c$  so long as  $(g^{s'} h^{r'} / c)^{l_G} = 1$ . The proofs above similarly goes through considering  $l_G$ -th power of any element in  $G$  belongs to  $\langle h \rangle$ .

To make verifiable encryption, we use the Okamoto-Uchiyama encryption [19]. Suppose that  $p, q$  are primes such that  $(p-1)/2, (q-1)/2$  are  $C$ -rough. Define  $\text{QR}(X) \triangleq \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} \text{ s.t. } y^2 = x \pmod{X}\}$ . Let  $H = \text{QR}(n) \subset G = \text{QR}(q) \subset (\mathbb{Z}/n\mathbb{Z})^\times$  where  $n = p^2 q$ . Since  $(\frac{x}{n}) = (\frac{x}{p})^2 (\frac{x}{q})$ , one can efficiently check that  $x$  belongs to  $G$  by computing the Jacobian symbols over  $n$ . Then  $g, h$  are chosen as follows: Pick up at random  $h_0 \in H$  such that  $p \nmid \text{ord}(h)$ . Set  $g = h_0^\alpha$  and  $h = h_0^n$ . We have  $l_G = 4$ . The commitment  $c = g^x h^r$  ( $0 < x < p$ ) is not statistical hiding but one can still think of it as computational hiding (so long as the OU encryption is semantically secure). The associated protocols above can be applied to this new commitment without any modification, which makes this verifiable. Actually in this case, the verifier can be convinced that the committer can only open commitments that belong to  $H$  (because if the committer can show any non-trivial  $\mu$  such that  $\mu^4 = 1$ , he can factor  $n$ ).

An application of this verifiable encryption appears in [13]. A “light” version of this verifiable encryption, using  $c = g^s \pmod{n}$ , appears in [11].

# Efficient Oblivious Transfer in the Bounded-Storage Model

Dowon Hong, Ku-Young Chang, and Heuisu Ryu

Information Security Research Division, ETRI,  
161 Gajeong-Dong, Yuseong-Gu, Daejeon, 305-350, Korea,  
{dwhong, jang1090, hsryu}@etri.re.kr

**Abstract.** In this paper we propose an efficient  $OT_1^N$  scheme in the bounded storage model, which is provably secure without complexity assumptions. Under the assumption that a public random string of  $M$  bits is broadcasted, the protocol is secure against any computationally unbounded dishonest receiver who can store  $\tau M$  bits,  $\tau < 1$ . The protocol requires the sender and the receiver to store  $N \cdot O(\sqrt{kM})$  bits, where  $k$  is a security parameter. When  $N = 2$ , our protocol is similar to that of Ding [10] but has more efficient round and communication complexities. Moreover, in case of  $N > 2$ , if the sender and receiver can store  $N \cdot O(\sqrt{kM})$  bits, we are able to construct a protocol for  $OT_1^N$  which has almost the same complexity as in  $OT_1^2$  scheme. Ding's protocol was constructed by using the interactive hashing protocol which is introduced by Noar, Ostrovsky, Venkatesan and Yung [15] with very large round-complexity. We propose an efficiently extended interactive hashing and analyze its security. This protocol answers partially an open problem raised in [10].

## 1 Introduction

Consider two parties of the sender Alice and the receiver Bob. Alice has  $N$  secret bits  $X_0, X_1, \dots, X_{N-1} \in GF(2)$ , and Bob has a secret value  $c \in \{0, 1, \dots, N-1\}$ . Alice sends  $X_0, X_1, \dots, X_{N-1}$  in such a way that Bob receives  $X_c$ , but does not learn any information about other secrets  $X_i$ ,  $i \neq c$ , and Alice learns nothing about  $c$ . An 1-out-of- $N$  Oblivious Transfer ( $OT_1^N$ ) is a cryptographic two-party protocol that provides a solution for the goal.

$OT_1^2$  was suggested by Even, Goldreich, and Lempel [11], as a generalization of Rabin's Oblivious Transfer (OT) [16], and Crépeau [6] proved that  $OT$  and  $OT_1^2$  are equivalent.  $OT_1^N$  was introduced by Brassard, Crépeau, and Robert [2] under the name ANDOS (all or nothing disclosure of secrets). Oblivious transfer can be used to construct cryptographic protocols, such as bit commitment, zero-knowledge proof, and generally secure multi-party computation [13, 21, 12, 7, 14].

Traditionally, oblivious transfer has been constructed under complexity assumptions, such as the hardness of factoring or discrete log, or the existence of trapdoor one-way permutations. However, they do not guarantee information-theoretic security, and the security of the protocol could be subverted later,

when enabled by breakthroughs in computing technology and algorithms. For example, protocols based on the hardness of factoring or computing discrete logarithms will become insecure if quantum computers become available [18]. Alternatives to computational security assumptions that have been proposed include quantum cryptography, the noisy channel model, and the bounded-storage model [18, 3].

Cachin, Crépeau, and Maril [4] proposed the first protocol for  $OT_1^2$  in the bounded-storage model that is unconditionally secure, without any complexity assumption. Under the assumption that a public random string of  $M$  bits is broadcasted, the CCM protocol [4] guarantees provable security against any computationally unbounded dishonest receiver who can store  $\tau M$  bits,  $\tau < 1$ . Furthermore, the security against a dishonest receiver is preserved regardless of future increases in storage capacity. The case where the storage bound is placed on the sender is equivalent by the reversibility of OT [9]. Protocols in the bounded-storage model make use of a very large amount of auxiliary information, called public random string [17], in order to defeat the adversary. The public random string could be a random bit sequence broadcasted by a satellite or transmitted between the legitimate parties, or the signal of a deep-space radio source. Recently, Ding [10] proposed a similar but more efficient protocol for  $OT_1^2$  in the bounded-storage model than the CCM protocol. Ding's protocol reduced the storage requirement from  $O(M^{2/3})$  in the CCM protocol, to  $O(\sqrt{kM})$  where  $k$  is a security parameter and proved that any dishonest receiver who stores  $O(M)$  bits succeeded with probability at most  $2^{-O(k)}$ , rather than inverse polynomially small.

In this paper, we propose a provably secure and efficient protocol for  $OT_1^N$  with a storage-bounded receiver, without any complexity assumption. Our protocol uses  $N$  public random strings of  $M$  bits and requires the sender and the receiver to store  $N \cdot O(\sqrt{kM})$  bits, where  $k$  is a security parameter. When  $N = 2$ , our protocol is similar to that of Ding's protocol but has more efficient round and communication complexities. Moreover, in case of  $N > 2$ , if the sender and the receiver can store  $N \cdot O(\sqrt{kM})$  bits, we are able to construct a protocol for  $OT_1^N$  which has almost the same complexity as in  $OT_1^2$  scheme. This is constructed based on an *extended interactive hashing* scheme.

Noar, Ostrovsky, Venkatesan and Yung [15] introduced the interactive hashing protocol, and Cachin, Crépeau, and Maril [4] gave a new elegant analysis on it. Interactive hashing is a protocol between a challenger Alice with no input and a responder Bob with input string  $\chi$  and provides a way to isolate two strings. One of the strings is Bob's input  $\chi$  and the other is chosen randomly, without influence from Bob. However, Alice does not learn that which one is  $\chi$ . Up to the present, the interactive hashing has been based on NOVY protocol [15] which has very large round and communication complexities. The round and communication complexities of NOVY protocol, which has the string of  $t$  bits to be transmitted, are  $t - 1$  rounds and  $t^2 - 1$  bits respectively. Thus Ding's protocol for  $OT_1^2$  which is based on NOVY protocol has very large round and communication complexities.

We propose more efficiently extended interactive hashing scheme than the NOVY protocol. We can accomplish the interactive hashing with  $t/m - 1$  rounds and  $t^2/m - m$  bits of communication complexity, when  $m$  is a divisor of  $t$ , and provide a way to isolate more than two strings. As a concrete example of what is claimed in this paper (Section 4), assume that the length of a public random string is one Petabit, (i.e.  $M = 10^{15}$ ), and  $1000 \leq k \leq 10000$  for a security parameter  $k$ , then we can choose  $k$  easily such that the protocol has  $t^{3/2} - t^{1/2}$ -bit communication complexity which is much lower than that of NOVY protocol. This result answers partially an open problem raised in [10].

This paper is organized as follows. In Section 2, we construct a new universal hash family. Using this, we propose an extended interactive hashing protocol. The protocol for  $OT_1^N$  in the bounded-storage model is presented in Section 3. In Section 4, we discuss the complexity of our protocol.

## 2 Extended Interactive Hashing

In this section we propose an efficiently extended interactive hashing protocol and give an analysis on it. In order to construct this, we first introduce a new universal hash family.

### 2.1 Universal Hash Family

The technique of universal hashing was introduced in 1979 by Carter and Wegman [5] and is used in many areas of computer science and cryptography [19,20].

**Definition 1.** Let  $\mathcal{F}$  be the set of all functions from  $X$  to  $Y$  and let  $\mathcal{H}$ -hash family be a subset of  $|\mathcal{H}|$  functions in  $\mathcal{F}$ .  $\mathcal{H}$ -hash family is called universal if, for any distinct elements  $x_1, x_2 \in X$ , there exist at most  $|\mathcal{H}|/|Y|$  functions  $h \in \mathcal{H}$  such that  $h(x_1) = h(x_2)$ .

Let  $t$  and  $m$  be positive integers such that  $m$  is a divisor of  $t$ . We now define a universal hash family from  $GF(2)^t$  to  $GF(2)^m$ . Let  $f(x)$  be an irreducible polynomial of degree  $m$  over  $GF(2)$ . Then  $GF(2^m) = GF(2)[x]/(f(x))$  is represented as  $\{\sum_{i=0}^{m-1} a_i x^i : a_i \in GF(2)\}$ . Define the bijective function  $\phi : GF(2)^m \rightarrow GF(2^m)$  by  $(a_{m-1}, \dots, a_1, a_0) \mapsto a_{m-1}x^{m-1} + \dots + a_1x + a_0$ . Let  $t = lm$ . Then  $GF(2)^t = (GF(2)^m)^l = \{(A_{l-1}, \dots, A_1, A_0) : A_i \in GF(2)^m, 0 \leq i \leq l-1\}$ . We regard  $GF(2)^t$  as  $(GF(2)^m)^l$  and let  $S = (GF(2)^m)^l$ . In order to define a universal hash family from  $S$  to  $GF(2)^m$ , for any  $\zeta = (\zeta_{l-1}, \dots, \zeta_1, \zeta_0) \in S$ , we define the hash function using the above function  $\phi$  as follows ;

$$h_\zeta : S \longrightarrow GF(2)^m$$

$$(A_{l-1}, \dots, A_1, A_0) \longmapsto \phi^{-1}\left(\sum_{i=0}^{l-1} \phi(A_i) \cdot \phi(\zeta_i)\right). \quad (1)$$

Consider the set  $\mathcal{H}$  of hash functions from  $S$  to  $GF(2)^m$  as follows ;

$$\mathcal{H} = \{h_\zeta : \zeta = (\zeta_{l-1}, \dots, \zeta_1, \zeta_0) \in S\},$$

where  $h_\zeta$  is defined in (1).

**Lemma 1.**  $\mathcal{H}$  is a universal hash family.

*Proof.* For any two distinct elements  $x = (x_{l-1}, \dots, x_0), y = (y_{l-1}, \dots, y_0) \in S$ , we need to count the number of  $\zeta = (\zeta_{l-1}, \dots, \zeta_0) \in S$  with  $h_\zeta(x) = h_\zeta(y)$ . Since  $x \neq y$ , there is an index  $i_0 \in \{0, \dots, l-1\}$  such that  $x_{i_0} \neq y_{i_0} \in GF(2)^m$ . Then for any  $\zeta \in S$

$$\begin{aligned} h_\zeta(x) = h_\zeta(y) &\Leftrightarrow \phi(\zeta_{i_0})(\phi(y_{i_0}) - \phi(x_{i_0})) + \sum_{i \neq i_0} \phi(\zeta_i)(\phi(y_i) - \phi(x_i)) = 0 \\ &\Leftrightarrow \phi(\zeta_{i_0})(\phi(y_{i_0}) - \phi(x_{i_0})) = \sum_{i \neq i_0} \phi(\zeta_i)(\phi(y_i) - \phi(x_i)) \in GF(2^m) \\ &\Leftrightarrow \phi(\zeta_{i_0}) = \sum_{i \neq i_0} \phi(\zeta_i)(\phi(y_i) - \phi(x_i)) \cdot (\phi(y_{i_0}) - \phi(x_{i_0}))^{-1}. \quad (2) \end{aligned}$$

Since  $\phi$  is bijective, for each choice of  $\zeta_i$ 's for  $i \neq i_0$ , equation (2) has exactly one solution in  $\zeta_{i_0}$ . Since the number of  $i$ 's for  $i \neq i_0$  is  $l-1$  and  $\zeta_i \in GF(2)^m$  for each  $i$ , there are exactly  $2^{m(l-1)} = |\mathcal{H}|/2^m$  functions  $h_\zeta \in \mathcal{H}$  with  $h_\zeta(x) = h_\zeta(y)$ . Thus,  $\mathcal{H}$  is a universal hash family.  $\square$

The universal hash family  $\mathcal{H}$  defined above has the following properties.

**Lemma 2.** Let  $\mathcal{H}$  be the hash family defined above. For any two nonzero distinct elements  $x, y \in S$  and for any  $b \in GF(2)^m$ , let  $T_b = \{h \in \mathcal{H} : h(x) = b, h(y) = b\}$ . Then  $|T_b| = |\mathcal{H}|/2^{2m}$ .

*Proof.* For any two nonzero elements  $x = (x_{l-1}, \dots, x_0), y = (y_{l-1}, \dots, y_0) \in S$ , let  $x \neq y$ . Note that  $T_b = \{\zeta \in S : h_\zeta(x) = b, h_\zeta(y) = b\}$  by definition. Since  $x \neq y$ , there are two distinct indices  $j, k \in \{0, \dots, l-1\}$  such that  $x_j \neq 0, y_k \neq 0 \in GF(2)^m$ . Then for any  $\zeta = (\zeta_{l-1}, \dots, \zeta_0) \in S$

$$\begin{aligned} h_\zeta(x) = b &\Leftrightarrow \phi(x_j)\phi(\zeta_j) + \sum_{i \neq j} \phi(x_i)\phi(\zeta_i) = \phi(b) \\ &\Leftrightarrow \phi(\zeta_j) = \left( \sum_{i \neq j} \phi(x_i)\phi(\zeta_i) + \phi(b) \right) \cdot \phi(x_j)^{-1}, \\ h_\zeta(y) = b &\Leftrightarrow \phi(\zeta_k) = \left( \sum_{i \neq k} \phi(y_i)\phi(\zeta_i) + \phi(b) \right) \cdot \phi(y_k)^{-1}. \end{aligned}$$

Hence by similar method in Lemma 1,  $|T_b| = 2^{m(l-2)} = |\mathcal{H}|/2^{2m}$ .  $\square$

**Lemma 3.** Let  $\mathcal{H}$  be the hash family defined above. Then for any nonzero element  $s \in S$  and for any  $b \in GF(2)^m$ ,  $|\{h \in \mathcal{H} : h(s) = b\}| = |\mathcal{H}|/2^m$ .

*Proof.* clear.  $\square$

## 2.2 Interactive Hashing

Interactive hashing is a two-party protocol between a challenger Alice and a responder Bob. Cachin, Crépeau, and Marcil [4] gave a new elegant analysis on it in order to be used to construct OT in the bounded-storage model. Bob has a secret  $t$ -bit string  $\chi \in T \subset GF(2)^t$ , where  $|T| \leq 2^{t-k}$  and  $\chi$  and  $T$  are unknown to Alice. At the end of the protocol, Alice receives two strings, one of which is  $\chi$ , but Alice does not know which one is  $\chi$ . Also, Bob cannot force both two strings to be in  $T$ , except with a small probability  $\nu(k)$ .

The following interactive hashing protocol is proposed in Noar, Ostrovsky, Venkatesan and Yung [15]: Alice randomly chooses  $t - 1$  linearly independent vectors  $a_1, \dots, a_{t-1} \in GF(2)^t$ . The protocol then proceeds in  $t - 1$  rounds. In Round  $i$ , for each  $i = 1, \dots, t - 1$ ,

1. Alice announces  $a_i$  to Bob.
2. Bob computes  $b_i = a_i \cdot \chi$  and sends  $b_i$  to Alice.

At the end, both Alice and Bob have the same system of linear equations  $b_i = a_i \cdot \chi$ ,  $i = 1, \dots, t - 1$  over  $GF(2)$ . Since  $a_1, \dots, a_{t-1} \in GF(2)^t$  are linearly independent, the system has exactly two  $t$ -bit strings  $\chi_1, \chi_2$  as solutions and one of them is  $\chi$  by standard linear algebra. Thus Alice does not know information-theoretically that which solution is  $\chi$ . Also, the condition that Bob cannot force both two strings to be in  $T$ , except with a small probability  $\nu(k)$ , was proved in [4].

Since the round and communication complexities of NOVY protocol, which transmits the string of  $t$  bits, are  $t - 1$  rounds and  $t^2 - 1$  bits respectively, the protocol which is based on NOVY protocol has very large round and communication complexities.

## 2.3 Extended Interactive Hashing Protocol

We propose a new scheme between a challenger Alice with no input and a responder Bob with input string  $\chi$  which provides a way to isolate more than two strings. Bob has a secret  $t$ -bit string  $\chi \in T \subset GF(2)^t$ , where  $|T| \leq 2^{t-k}$  and  $\chi$  and  $T$  are unknown to Alice. For some positive integers  $l$  and  $m$ , let  $t = lm$ . The protocol should meet the following requirements:

1. Bob sends a secret  $t$ -bit string in such a way that Alice receives  $2^m$   $t$ -bit strings and one of them is  $\chi$ , but Alice does not know that which one is  $\chi$ .
2. Bob cannot force any two of them to be in  $T$ , except with a small probability  $\nu(k)$ .

We regard  $GF(2)^t$  as  $(GF(2)^m)^l$  and let  $S = (GF(2)^m)^l$ . Bob chooses a secret  $t$ -bit string  $\chi = (\chi_{l-1}, \dots, \chi_1, \chi_0) \in S$ , where  $\chi_i \in GF(2)^m$ ,  $0 \leq i \leq l - 1$ . Now we consider the universal family  $\mathcal{H}$  of hash functions from  $S$  to  $GF(2)^m$  which is defined in Section 2.1 as

$$\mathcal{H} = \{h_\zeta : \zeta = (\zeta_{l-1}, \dots, \zeta_1, \zeta_0) \in S\},$$

where  $h_\zeta$  is defined in (1).

Our scheme is described below.

**Protocol :** The protocol operates in  $t/m - 1$  rounds. In Round  $i$ , for  $i = 1, \dots, t/m - 1$ ,

1. Alice chooses a function  $h_i \in \mathcal{H}$  with uniform distribution. Let  $a_i \in GF(2)^t$  be the description vector of  $h_i$  such that  $h_i = h_{a_i}$ . If  $a_i$  is linearly dependent in  $a_1, \dots, a_{i-1}$ , then Alice repeats this step until it is independent. Alice sends  $a_i$  to Bob.
2. Let  $a_i = (a_i^{(t/m-1)}, \dots, a_i^{(1)}, a_i^{(0)}) \in S$ ,  $a_i^{(j)} \in GF(2)^m$ ,  $0 \leq j \leq t/m - 1$ . Bob computes  $m$ -bit  $b_i = h_{a_i}(\chi) = \phi^{-1} \left( \sum_{j=0}^{t/m-1} \phi(a_i^{(j)}) \cdot \phi(\chi_j) \right)$ , and sends  $b_i$  to Alice.

After the  $t/m - 1$  rounds, both Alice and Bob have the same  $t/m - 1$  linear equations over  $GF(2)^m$  with  $\chi$  as a solution. The system has exactly  $2^m$   $t$ -bit strings  $\chi_0, \dots, \chi_{2^m-1}$  as solutions, one of which is  $\chi$ . We call this scheme *extended interactive hashing*. We note that in case of  $m = 1$ , our protocol is the same as interactive hashing.

It is clear that Alice does not know information-theoretically that which solution is  $\chi$ . Thus Condition 1 of extended interactive hashing is satisfied. We now come to Condition 2 regarding the security against a dishonest responder Bob. In our protocol, Bob can cheat if he can answer Alice's queries in such a way that  $T$  contains two distinct elements  $s_1, s_2$  received by Alice. In Theorem 1, we show that Bob can only cheat in extended interactive hashing if the size of  $|T|$  is close to  $|GF(2)^t| = 2^t$ . In order to prove this, we need some lemmas.

The following lemma shows that each round of scheme reduces the size of  $T$  by a factor of almost  $2^m$  with very high probability. This approach was used first to prove the security of interactive hashing in [4]. We improve this method in our model.

**Lemma 4.** *Let  $T \subset GF(2)^t$  be any subset with  $|T| = 2^{\alpha t}$  for  $0 < \alpha < 1$  and let  $p$  be a positive integer such that  $p \leq \alpha t/3$ . Let  $m$  be a positive integer which is a divisor of  $t$ . Let  $\mathcal{H}$  be the universal family of hash functions from  $GF(2)^t$  to  $GF(2)^m$  defined above. Let  $U$  be a random variable with uniform distribution over  $\mathcal{H}$ . Then for any  $b \in GF(2)^m$ ,*

$$\Pr \left[ |\{s \in T : U(s) = b\}| < \left( \frac{1}{2^m} + \frac{1}{2^{p+m/2}} + \frac{1}{2^{3p}} \right) |T| \right] \geq 1 - 2^{-p}.$$

*Proof.* For any  $s \in T$  and  $b \in GF(2)^m$ , we consider the following random variables

$$X_{(b,s)} = \begin{cases} 1 & \text{if } U(s) = b \\ 0 & \text{otherwise} \end{cases}$$

and their sum  $X_b = \sum_{s \in T} X_{(b,s)} = |\{s \in T : U(s) = b\}|$ . Thus we must show that for any  $b \in GF(2)^m$ ,

$$\Pr \left[ X_b < \left( \frac{1}{2^m} + \frac{1}{2^{p+m/2}} + \frac{1}{2^{3p}} \right) |T| \right] \geq 1 - 2^{-p}. \quad (3)$$



Case 1:  $b \neq 0 \in GF(2)^m$ .

By the definition  $X_{(b,s)}$  and Lemma 3, we obtain that for any  $s \neq 0 \in T$

$$\begin{aligned} E[X_{(b,s)}] &= E[X_{(b,s)}^2] = 1 \cdot \frac{|\{h \in \mathcal{H} : h(s) = b\}|}{|\mathcal{H}|} \\ &= \frac{1}{2^m}, \end{aligned}$$

and  $X_{(b,0)} = X_{(b,0)}^2 = 0$  by the definition of our hash family. Thus  $E[X_b] = \frac{|T|-1}{2^m}$ . By the definition of  $X_b$ , we obtain that

$$E[X_b^2] = E\left[\sum_{s \in T} X_{(b,s)}^2 + 2 \sum_{s_i < s_j \in T} X_{(b,s_i)} X_{(b,s_j)}\right].$$

Since  $b \neq 0$ ,  $X_{(b,0)} = 0$ . Using this fact and Lemma 2 we obtain that

$$\begin{aligned} E\left[\sum_{s_i < s_j \in T} X_{(b,s_i)} X_{(b,s_j)}\right] &= \sum_{0 < s_i < s_j \in T} E[X_{(b,s_i)} X_{(b,s_j)}] \\ &= \sum_{0 < s_i < s_j \in T} \frac{|\{h \in \mathcal{H} : h(s_i) = h(s_j) = b\}|}{|\mathcal{H}|} \\ &< \frac{(|T|-1)^2}{2} \cdot \left(\frac{1}{2^m}\right)^2. \end{aligned}$$

Thus, we have  $E[X_b^2] < \frac{|T|-1}{2^m} + \left(\frac{|T|-1}{2^m}\right)^2$  and

$$\begin{aligned} Var[X_b] &= E[X_b^2] - (E[X_b])^2 \\ &< \frac{|T|-1}{2^m}. \end{aligned}$$

Now, by Chebychev Inequality we obtain that for any  $b \neq 0 \in GF(2)^m$  and  $\delta > 0$

$$\Pr\left[\left|X_b - \frac{|T|-1}{2^m}\right| \geq \delta\right] < \frac{|T|-1}{2^m \delta^2}.$$

Substituting  $\delta = \sqrt{2^p(|T|-1)/2^m}$ , we have

$$\Pr\left[\left|X_b - \frac{|T|-1}{2^m}\right| \geq 2^{\frac{p+\alpha t-m}{2}}\right] < 2^{-p}.$$

Hence, if  $p \leq \alpha t/3$ , then with probability at least  $1 - 2^{-p}$ , we obtain

$$\begin{aligned} X_b &< \left(\frac{1}{2^m} + 2^{\frac{p+\alpha t-m}{2}-\alpha t}\right) |T| \\ &< \left(\frac{1}{2^m} + \frac{1}{2^{p+m/2}}\right) |T| \end{aligned}$$

and (3) is satisfied.

Case 2:  $b = 0 \in GF(2)^m$ .

Using  $X_{(0,0)} = 1$ , Lemma 2 and Lemma 3, we obtain that  $E[X_0] = \frac{|T|-1}{2^m} + 1$  and  $E[X_0^2] < \frac{3(|T|-1)}{2^m} + 1 + \left(\frac{|T|-1}{2^m}\right)^2$ . Thus  $Var[X_0] < \frac{|T|-1}{2^m}$ . By Chebychev Inequality we obtain that for any  $\delta > 0$

$$\Pr \left[ \left| X_0 - \left( \frac{|T|-1}{2^m} + 1 \right) \right| \geq \delta \right] < \frac{|T|-1}{2^m \delta^2}.$$

Substituting  $\delta = \sqrt{2^p(|T|-1)/2^m}$ , we have that with probability at least  $1-2^{-p}$ ,

$$X_0 < \left( \frac{1}{2^m} + \frac{1}{2^{p+m/2}} + \frac{1}{2^{3p}} \right) |T|$$

using  $p \leq \alpha t/3$ , and the lemma is proved.  $\square$

The following lemma was proved in [4]

**Lemma 5.** [4] *Let  $T \subset GF(2)^t$  be any subset with  $|T| = 2^{\alpha t}$  for  $0 < \alpha < 1$ . Let  $p$  and  $q$  be positive integers such that  $2\alpha t < mq - p$  and  $p, mq \leq t$  where  $m$  is a divisor of  $t$ . Let  $\mathcal{H}$  be the universal family of hash functions from  $GF(2)^t$  to  $GF(2)^{mq}$ . Let  $U$  be a random variable with uniform distribution over  $\mathcal{H}$ . Then for any distinct  $s_1, s_2 \in T$ , we have*

$$\Pr[U(s_1) = U(s_2)] \leq 2^{-p}.$$

**Lemma 6.** *Suppose that Alice and Bob engage in extended interactive hashing of a  $t$ -bit string as described above. Let  $T \subset GF(2)^t$  be any subset with  $|T| = 2^{\alpha t}$  for  $0 < \alpha < 1$  and let  $r$  be a positive integer such that  $\log_2 t \leq r \leq \alpha t/6$ . Let  $m$  be a positive integer which is a divisor of  $t$  and  $m \leq 2r$ . If  $\alpha < 1 - \frac{8r+2m+2}{t}$ , then with probability at most  $\frac{1}{m2^r}$ , Bob can answer Alice's queries in such a way that Bob's answers are consistent for two distinct elements  $s_1, s_2 \in T$ .*

*Proof.* For  $i = 1, \dots, t/m-1$ , let  $T_i \subset T$  be the subset of  $T$  satisfying  $h_j(s) = b_j$ , for  $j = 1, \dots, i$ , after Round  $i$  of the extended interactive hashing protocol. Let  $p = 2r$ . Then using  $r \leq \alpha t/6$  and  $\alpha < 1 - \frac{8r+2m+2}{t}$ , we obtain that  $\alpha t \geq 3p$  and  $\frac{\alpha t - 3p}{m} + 1 < \frac{t}{m} - 1$ . Thus there exists a positive integer  $i_j$  such that

$$0 \leq \frac{\alpha t - 3p}{m} < i_j \leq \frac{\alpha t - 3p}{m} + 1 < \frac{t}{m} - 1. \quad (4)$$

Applying Lemma 4 by induction on  $i$  from 1 to  $i_j - 1$ , we get

$$|T_i| < \left( \frac{1}{2^m} + \frac{1}{2^{p+m/2}} + \frac{1}{2^{3p}} \right)^i |T|,$$

except with probability at most  $i \cdot 2^{-p}$ . Thus, we obtain that

$$\log_2 |T_{i_j}| < (\alpha t - m i_j) + i_j \log_2(1 + 2^{m/2-p} + 2^{-3p+m}) < 3p + 1 \quad (5)$$

by (4) and  $i_j \log_2(1 + 2^{m/2-p} + 2^{-3p+m}) < t/m \cdot (2^{m/2-p} + 2^{-3p+m}) < 1$ .

Now we want to apply Lemma 5 for step  $i_j$  (round  $i_j$  through  $t/m - 1$  collectively) using  $T_{i_j}$ . Since  $\alpha < 1 - \frac{4p+2m+2}{t}$ ,  $4p < t - \alpha t - 2m - 2$  and  $2\log_2 |T_{i_j}| < 6p + 2 < 2p + t - \alpha t - 2m$  by (5). Using (4) we get

$$2p + t - \alpha t - 2m = t - m \left( \frac{\alpha t - 3p}{m} + 2 \right) - p \leq t + m(-i_j - 1) - p$$

and  $2\log_2 |T_{i_j}| < m(t/m - 1 - i_j) - p$  holds. Hence we can apply Lemma 5 and the overall failure probability is at most  $(i_j + 1)2^{-p} < t/m \cdot 2^{-p} < \frac{1}{m2^r}$ , which proves the lemma.  $\square$

The following theorem shows that Condition 2 of extended interactive hashing is satisfied.

**Theorem 1.** *Suppose that Alice and Bob engage in extended interactive hashing of a  $t$ -bit string as defined above. For positive integers  $l$  and  $m$ , let  $t = lm$ . Let  $T \subset GF(2)^t$  be any subset with  $|T| \leq 2^{t-k}$  where  $k$  satisfies  $\log_2 t \leq k \leq 2t/3$ . If  $m < \frac{k-2}{6}$ , then with probability at most  $\frac{2^{-O(k)}}{m}$ , Bob can answer Alice's queries in such a way that Bob's answers are consistent for two distinct elements  $s_1, s_2 \in T$ .*

*Proof.* For any positive integer  $r$  which satisfies  $\log_2 t \leq r \leq (t-2)/18$ , let  $k = 12r + 2$ . Then we get  $r \leq \frac{t-k}{6}$  and  $m < 2r$ . Thus the theorem follows from Lemma 6.  $\square$

**Corollary 1.** *Suppose that Alice and Bob engage in extended interactive hashing of a  $t$ -bit string as defined above. For positive integers  $l$  and  $m$ , let  $t = lm, m < t$ . Let  $T_0, T_1 \subset GF(2)^t$  be any two subsets with  $|T_0|, |T_1| \leq 2^{t-k}$  where  $k$  satisfies  $\log_2 t \leq k \leq 2t/3$ . If  $m < \frac{k-2}{6}$ , then the probability that Bob can answer Alice's queries such that two distinct elements, which one lies in  $T_0$  and the other one lies in  $T_1$ , are consistent with his answers is at most  $\frac{2^{-O(k)}}{m}$ .*

### 3 1-out of- $N$ Oblivious Transfer Protocol

In this section we describe an efficient protocol for  $OT_1^N$  in the bounded-storage model. Throughout the paper, let  $k$  be a security parameter and  $M$  be the length of a public random string, and let  $L = \tau M$ ,  $\tau < 1$ , be the storage bound on the receiver Bob. For simplicity, we only consider  $L = M/6$  (i.e.  $\tau = 1/6$ ). For any  $\tau < 1$  we can obtain similar results.

An  $OT_1^N$  scheme is a two-party protocol between the sender Alice who possesses  $N$  secret bits  $X_0, \dots, X_{N-1} \in GF(2)$  and the receiver Bob who would like to learn one of them at his choice. We assume that Alice is honest, that is, it won't send secrets that are not claimed. An  $OT_1^N$  scheme should satisfy the following requirements:

1. **Correctness:** if Alice and Bob follow the protocol, Bob obtains  $X_c$  after executing the protocol, where  $c \in \{0, \dots, N-1\}$  is a secret value of his choice.

2. Bob's privacy: after executing the protocol with Bob, Alice shall not get any information about Bob's secret value  $c$ .
3. Alice's privacy: after executing the protocol with Alice, Bob does not learn any information about other secrets  $X_i, i \neq c$  or their combination except with a negligible probability  $\nu(k)$ .

### 3.1 Basis Ideas

In this subsection we explain the basic ideas of our protocol for  $OT_1^N$ . Let  $n = 2\sqrt{kM}$ .

First, Alice and Bob choose independent random subsets  $\mathcal{A}, \mathcal{B} \subset \{1, \dots, M\}$  with  $|\mathcal{A}| = |\mathcal{B}| = n$ , respectively. If public random string  $\alpha \xleftarrow{R} GF(2)^M$  is broadcasted, Alice stores  $\alpha[i], \forall i \in \mathcal{A}$  and Bob stores  $\alpha[j], \forall j \in \mathcal{B}$ , where  $\alpha[i]$  is the  $i$ -th bit of  $\alpha$ . Then Alice sends her subset  $\mathcal{A}$  to Bob, and Bob computes  $\mathcal{A} \cap \mathcal{B}$ . Following lemma shows that  $|\mathcal{A} \cap \mathcal{B}| \geq k$  with very high probability.

**Lemma 7.** [10] Let  $\mathcal{A}, \mathcal{B}$  be two independent random subset of  $\{1, \dots, M\}$  with  $|\mathcal{A}| = |\mathcal{B}| = 2\sqrt{kM}$ . Then  $\Pr[|\mathcal{A} \cap \mathcal{B}| < k] < e^{-k/4}$ .

**Fact 1. (Encoding  $k$ -Element Subsets)** [4]. Each of the  $\binom{n}{k}$   $k$ -element subsets of  $\{1, \dots, n\}$  can be uniquely encoded as a  $\lceil \log_2 \binom{n}{k} \rceil$ -bit string.

Next, Bob encodes a random  $k$ -element subset  $\mathcal{A}_I \subset \mathcal{A} \cap \mathcal{B}$  as a  $\lceil \log_2 \binom{n}{k} \rceil$ -bit string and sends  $\mathcal{A}_I$  to Alice by the extended interactive hashing protocol defined in Section 2.3. After executing the extended interactive hashing protocol between Alice and Bob, they can construct one “good” set and  $N-1$  “bad” sets. Bob knows the “good” set, but does not learn any information about the “bad” sets. Alice knows all of the sets, but does not distinguish between the “good” set and the “bad” sets.

Next, Bob asks Alice to encrypt  $X_c$  with the “good” set and other secrets  $X_i, i \neq c$  with the bad sets. Since Bob knows the “good” set, not the “bad” sets, he can recover  $X_c$ , but not  $X_i \neq c$ .

### 3.2 Protocol for $OT_1^N$

We propose the  $OT_1^N$  protocol for a receiver with bounded memory size. The protocol uses  $N$  public random string  $\alpha_0, \dots, \alpha_{N-1} \xleftarrow{R} GF(2)^M$ . Let  $n = 2\sqrt{kM}$  and let  $t = \lceil \log_2 \binom{n}{k} \rceil$ . For some positive integers  $l$  and  $m$ , suppose  $t = lm$  and  $m < (k-2)/6$ .

**Protocol ( $OT_1^N$ ):** A sender Alice has  $N$  input bits  $X_0, \dots, X_{N-1}$  when  $N = 2^u, 1 \leq u \leq m$ . A receiver Bob chooses  $c \in \{0, \dots, N-1\}$  and want to know  $X_c$ .

1. Alice randomly chooses  $N$  sets  $\mathcal{A}^{(0)} = \{a_1^{(0)}, \dots, a_n^{(0)}\}, \dots, \mathcal{A}^{(N-1)} = \{a_1^{(N-1)}, \dots, a_n^{(N-1)}\} \subset \{1, \dots, M\}$  with length  $n$ . Bob randomly chooses  $N$  sets  $\mathcal{B}^{(0)} = \{b_1^{(0)}, \dots, b_n^{(0)}\}, \dots, \mathcal{B}^{(N-1)} = \{b_1^{(N-1)}, \dots, b_n^{(N-1)}\} \subset \{1, \dots, M\}$  with length  $n$ .

2. If the first public random string  $\alpha_0 \xleftarrow{R} GF(2)^M$  is broadcasted, Alice stores  $\alpha_0[a_1^{(0)}], \dots, \alpha_0[a_n^{(0)}]$  and Bob stores  $\alpha_0[b_1^{(0)}], \dots, \alpha_0[b_n^{(0)}]$ . After a short time, if the second public random string  $\alpha_1 \xleftarrow{R} GF(2)^M$  is broadcasted, Alice stores  $\alpha_1[a_1^{(1)}], \dots, \alpha_1[a_n^{(1)}]$  and Bob stores  $\alpha_1[b_1^{(1)}], \dots, \alpha_1[b_n^{(1)}]$ . After iterative procedures, if  $\alpha_{N-1} \xleftarrow{R} GF(2)^M$  is broadcasted, Alice stores the  $\alpha_{N-1}[a_1^{(N-1)}], \dots, \alpha_{N-1}[a_n^{(N-1)}]$  and Bob also stores the  $\alpha_{N-1}[b_1^{(N-1)}], \dots, \alpha_{N-1}[b_n^{(N-1)}]$ .
3. Alice sends  $\mathcal{A}^{(0)}, \dots, \mathcal{A}^{(N-1)}$  to Bob. Bob randomly chooses  $\varepsilon \xleftarrow{R} \{0, \dots, N-1\}$ , and computes  $\mathcal{A}^{(\varepsilon)} \cap \mathcal{B}^{(\varepsilon)}$ . If  $|\mathcal{A}^{(\varepsilon)} \cap \mathcal{B}^{(\varepsilon)}| < k$ , then he aborts the protocol. Otherwise, Bob chooses a set  $I = \{i_1, \dots, i_k\}$  such that  $\mathcal{A}_I^{(\varepsilon)} = \{a_{i_1}^{(\varepsilon)}, \dots, a_{i_k}^{(\varepsilon)}\} \subset \mathcal{A}^{(\varepsilon)} \cap \mathcal{B}^{(\varepsilon)}$ .
4. Bob encodes  $I$  as a  $t$ -bit string, where  $t = \lceil \log_2 \binom{n}{k} \rceil$ . Bob sends  $I$  to Alice with the extended interactive hashing protocol in  $t/m - 1$  rounds. After executing the extended interactive hashing, both Alice and Bob have exactly  $2^m$   $t$ -bit strings, one of which is  $I$ . Bob chooses  $N$  subsets  $I_0 < \dots < I_{N-1}$  such that  $I = I_\delta$  for some  $\delta \in \{0, \dots, N-1\}$  and such that  $N$  strings that encode  $I_0, \dots, I_{N-1}$  are among the  $2^m$  possible strings from the extended interactive hashing protocol, and sends them to Alice.
5. Alice checks whether  $N$   $k$ -subsets  $I_0 < \dots < I_{N-1} \subset \{1, \dots, n\}$  received in Step 4 are contained in all of  $2^m$   $k$ -subsets, computed by the extended interactive hashing protocol. If any one of  $N$   $k$ -subsets isn't contained in  $2^m$   $k$ -subsets, she aborts the protocol. For some  $\delta \in \{0, \dots, N-1\}$ ,  $I = I_\delta$ . Bob knows  $\delta$ , but Alice does not know  $\delta$ .
6. Bob sends  $u$  bits  $\gamma = \delta \oplus \varepsilon$  and  $\rho = c \oplus \varepsilon$  to Alice, where for any  $x, y \in \{0, \dots, N-1\}$ ,  $x \oplus y$  is defined as follows:  $x \oplus y = (x_0 \oplus y_0, \dots, x_{u-1} \oplus y_{u-1}) \in GF(2)^u$  where  $x = (x_0, \dots, x_{u-1}), y = (y_0, \dots, y_{u-1}) \in GF(2)^u$ .
7. Alice sets  $Y_0 = \bigoplus_{j=1}^k \alpha_0[a_{I_\gamma[j]}^{(0)}], \dots, Y_{N-1} = \bigoplus_{j=1}^k \alpha_{N-1}[a_{I_{\gamma \oplus N-1}[j]}^{(N-1)}]$  where  $I_l[j]$  denote the  $j$ -th element of  $k$ -subset  $I_l$ , for  $l = 0, \dots, N-1$ . Then Alice computes  $Z_0 = X_0 \oplus Y_\rho, \dots, Z_{N-1} = X_{N-1} \oplus Y_\rho \oplus Y_{N-1}$ , and sends  $Z_0, \dots, Z_{N-1}$  to Bob.
8. Bob gets  $X_c = Z_c \oplus Y_\varepsilon$ .

*Remark 1.* Alice and Bob store  $N \cdot n = 2N\sqrt{kM}$  bits in Step 2. Alice and Bob also store  $t^2/m$  bits in the extended interactive hashing of the Step 4. Here  $t = \lceil \log_2 \binom{n}{k} \rceil < k \cdot (\log_2 n - \log_2 k/e)$ . Because  $k \ll M$ , they need to store  $O(n)/m$  bits. Thus, in order to implement the protocol, Alice and Bob should store  $N \cdot n + O(n)/m$  bits.

*Remark 2.* The probability that an honest receiver Bob aborts in Step 3 of the protocol, is not more than  $e^{-k/4}$  by Lemma 6.

*Correctness:* Since  $Y_\varepsilon = \bigoplus_{j=1}^k \alpha_\varepsilon[a_{I_{\gamma \oplus \varepsilon}[j]}^{(\varepsilon)}] = \bigoplus_{j=1}^k \alpha_\varepsilon[a_{I[j]}^{(\varepsilon)}]$ , Bob can know  $Y_\varepsilon$ . Thus, he can compute  $X_c = Z_c \oplus Y_{\rho \oplus c} = Z_c \oplus Y_\varepsilon$ .

*Bob's Privacy:* Because Alice does not know  $\varepsilon$  defined in Step 3 and  $\delta$  defined in Step 5, She gains no information about the Bob's secret  $c$  with  $\gamma$  and  $\rho$  received from Step 6.

*Alice's Privacy:* In order to prove the security against a dishonest receiver Bob, who can store  $L = M/6$  bits, we apply the method of proof in the Ding's model [10]. If  $\alpha_0$  is broadcasted in Step 2, Bob computes an arbitrary function  $\eta_0 = A_0(\eta_0)$ ,  $|\eta_0| = M/6$  using unlimited computing power. And if  $\alpha_1$  is broadcasted, Bob computes an arbitrary function  $\eta_1 = A_1(\eta_0, \alpha_1)$ ,  $|\eta_1| = M/6$ . After iterative procedures, if  $\alpha_{N-1}$  is broadcasted, Bob computes an arbitrary function  $\eta_{N-1} = A_{N-1}(\eta_{N-2}, \alpha_{N-1})$ ,  $|\eta_{N-1}| = M/6$ . In Step 3 - Step 6, using  $\mathcal{A}^{(0)}, \dots, \mathcal{A}^{(N-1)}$  and  $\eta_{N-1}$ , Bob uses an arbitrary strategy in interacting with Alice. After executing the protocol, Bob tries to gain an information about  $X_i, i \neq c$ , using the information  $\eta_{N-1}$  on  $(\alpha_0, \dots, \alpha_{N-1})$ ,  $Z_0, \dots, Z_{N-1}$  received from Alice in Step 7, and all information  $\Omega$  which he gains in Step 3 - Step 6.

**Theorem 2.** *Consider the  $OT_1^N$  protocol defined above. For any  $A_0 : GF(2)^M \rightarrow GF(2)^{M/6}$ ,  $A_1 : GF(2)^{M/6} \times GF(2)^M \rightarrow GF(2)^{M/6}$ ,  $\dots$ ,  $A_{N-1} : GF(2)^{M/6} \times GF(2)^M \rightarrow GF(2)^{M/6}$ , for any strategy Bob uses in Step 3 - Step 6 of the protocol, with probability at least  $1 - 2^{-O(k)} - N \cdot 2^{-0.02M}$ , there exist some  $\rho \in \{0, 1, \dots, N-1\}$  such that  $\forall X_0, \dots, X_{N-1} \in GF(2)$ ,  $\forall c \in \{0, \dots, N-1\}$ ,  $\forall i \in \{1, \dots, N-1\}$  and for any distinguisher  $\mathcal{D}$ ,*

$$\begin{aligned} & | \Pr[ \mathcal{D}(\eta_{N-1}, \Omega, Y_{\rho \oplus i} \oplus X_c, Y_\rho \oplus X_{c \oplus i}) = 1 ] \\ & - \Pr[ \mathcal{D}(\eta_{N-1}, \Omega, Y_{\rho \oplus i} \oplus X_c, Y_\rho \oplus 1 \oplus X_{c \oplus i}) = 1 ] | < 2^{-k/3}, \end{aligned} \quad (6)$$

where  $\eta_0 = A_0(\alpha_0)$ ,  $\eta_1 = A_1(\eta_0, \alpha_1)$ ,  $\dots$ ,  $\eta_{N-1} = A_{N-1}(\eta_{N-2}, \alpha_{N-1})$ ,  $\Omega$  denotes all the information Bob obtains in Step 3 - Step 6, and  $Y_0, \dots, Y_{N-1}$  are defined in Step 7. Thus the view of Bob is essentially the same, even though  $X_{c \oplus i}$  is replaced by  $1 \oplus X_{c \oplus i}$ . Hence Bob gains no information about any non-trivial function or relation involving more than two  $X_i$ 's in the protocol.

A proof of this theorem which guarantees the privacy of Alice is given in the appendix.

## 4 Complexity

In the bounded-storage model, complex of  $OT_1^N$  mainly depends on the extended interactive hashing scheme. Since the complexity of the extended interactive hashing scheme for  $OT_1^N$  is similar to that of  $OT_1^2$ , we compare the complexity of our extended interactive hashing protocol for  $OT_1^2$  with the complexity of NOVY protocol, which is an interactive hashing scheme used in the CCM protocol [4] and Ding's protocol [10].

The NOVY protocol, which transmits the string of  $t$ -bits, has  $t-1$  rounds complexity and  $(t-1) \cdot (t+1) = t^2 - 1$  bits of communication complexity. On the other hand our extended interactive hashing protocol has  $t/m - 1$  rounds

**Table 1.**  $M = 10^{15}$ ,  $n = \lceil 2\sqrt{kM} \rceil$ ,  $t = \lceil \log_2 \binom{n}{k} \rceil$  and  $m_{max}$  is the largest positive integer  $m$ , which divides  $t$  and  $m < (k - 2)/6$ .

$k$	the number of $k$ such that	the number of $k$ such that
	$m_{max} \geq \sqrt{t}$	$m_{max} = 1$
1000 - 2000	218	101
2001 - 3000	329	100
3001 - 4000	353	92
4001 - 5000	389	95
5001 - 6000	403	90
6001 - 7000	414	77
7001 - 8000	440	75
8001 - 9000	426	93
9000 - 10000	445	65

complexity and  $(t/m - 1) \cdot (t + m) = t^2/m - m$  bits of communication complexity when  $m$  divides  $t$ . In case of  $m = 1$ , we note that our protocol and the NOVY protocol are same. If there exists  $m$  such that  $m > 1$ , our protocol can be constructed about  $m$  times as efficient as compared with the NOVY protocol. As  $m$  is large, we see that the complexity of our protocol is more reduced. By Theorem 1,  $m$  satisfies the following condition;  $1 \leq m < (k - 2)/6$ , where  $k$  is a security parameter. Thus if we choose the largest integer  $m$  such that  $m$  divides  $t$  and  $1 \leq m < (k - 2)/6$ , then we can obtain the integer  $m$  which makes our protocol most efficient. For example, assume that the length of public random string is Petabit (i.e.  $M = 10^{15}$ ) and  $1000 \leq k \leq 10000$  for a security parameter  $k$ . Table 1 gives the information for a security parameter  $k$  that we can choose in our protocol.

In case  $m_{max} = 1$  in Table 1, our interactive hashing protocol is simply equivalent to the NOVY protocol. By Table 1 we have that the number of  $k$  such that  $m_{max} = 1$  is less than 10% for  $1000 \leq k \leq 10000$ . If we choose  $k$  such that  $m_{max} \geq \sqrt{t}$ , then we can construct protocol which has much lower communication complexity of  $t^{3/2} - t^{1/2}$  bits than that of the NOVY protocol. Such  $k$  are more than 20% for  $1000 \leq k \leq 2000$ , 30% for  $2001 \leq k \leq 5000$  and 40% for  $5001 \leq k \leq 10000$ . Hence, we can choose  $k$  easily such that our extended interactive hashing for  $OT_1^2$  becomes more efficient than the NOVY protocol for CCM protocol and Ding's protocol.

## 5 Conclusion

In this paper we propose the  $OT_1^N$  protocol as a generalization of the Ding's protocol for  $OT_1^2$  in the bounded-storage model. Furthermore, when  $N = 2$ , our protocol is similar to that of Ding, but is constructed more efficient than that of Ding. We used the efficiently extended interactive hashing protocol for the sake of reducing a complexity of the protocol. The proposed extended interactive hashing protocol which transmits  $t$ -bit string has  $t/m - 1$  round complexity and

$(t/m - 1) \cdot (t + m) = t^2/m - m$  bits of communication complexity when  $m$  divides  $t$ , and provides a way to isolate more than two strings. We note that a given  $m$  in this paper must divide  $t$  and satisfy  $m < (k - 2)/6$ . And we show that we can choose an integer  $m$  such that the protocol has  $t^{3/2} - t^{1/2}$  bit communication complexity which is much lower than that of NOVY protocol by a concrete example. This fact provides a partial answer for an open problem raised in [10]. Using such extended interactive hashing, we also constructed the protocol for  $OT_1^N$  having almost the same efficiency as  $OT_1^2$  scheme.

## References

1. C. H. Bennett, G. Brassard, C. Crépeau, and M. H. Skubiszewska, *Practical quantum oblivious transfer protocols*, In Advances in Cryptology - CRYPTO '91, pp. 351-366, 1991.
2. G. Brassard, C. Crépeau, and J. M. Robert, *All-or-nothing disclosure of secrets*, In Advances in Cryptology - Crypto 86, pp. 234-238, 1987.
3. C. Cachin and U. Maurer, *Unconditional security against memory-bounded adversaries*, In Advances in Cryptology - CRYPTO '97, pp. 292-306, 1997.
4. C. Cachin, C. Crépeau, and J. Marcil, *Oblivious transfer with a memory-bounded receiver*, In Proc. 39th IEEE Symposium in Foundations of Computer Science, pp. 493-502, 1998.
5. J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences 18, pp. 143-154, 1979.
6. C. Crépeau, *Equivalence between two flavours of oblivious transfer*, In Advances in Cryptology - CRYPTO '87, pp. 351-368, 1987.
7. C. Crépeau, J. van de Graff, and A. Tapp, *Committed oblivious transfer and private multi-party computations*, In Advances in Cryptology - CRYPTO '95, pp. 110-123, 1995.
8. C. Crépeau and J. Kilian, *Achieving oblivious transfer using weakened security assumptions*, In Proc. 29th IEEE Symposium in the Foundations of Computer Science, pp. 42-52, 1988.
9. C. Crépeau and M. Sántha, *On the reversibility of oblivious transfer and private multi-party computations*, In Advances in Cryptology - CRYPTO '95, pp. 110-123, 1995.
10. Y. Z. Ding, *Oblivious Transfer in the Bounded Storage Model*, In Advances in Cryptology - CRYPTO 2001, pp. 155-170, 2001.
11. S. Even, O. Goldreich, and A. Lempel, *A randomized protocol for signing contracts*, In Advances in Cryptology - CRYPTO '82, pp. 205-210, 1982.
12. O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game or a completeness theorem for protocols with honest majority*, In Proc. 19th ACM Symposium on Theory of Computing, pp. 218-229, 1987.
13. J. Kilian, *Founding cryptography on oblivious transfer*, In Proc. 20th ACM Symposium on Theory of Computing, pp. 20-31, 1988.
14. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky, *Reducibility and completeness in private computations*, SIAM Journal on Computing, 29(4), pp.1189-1208, 2000.
15. M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung, *Perfect zero-knowledge arguments for NP using any one-way function*, Journal of Cryptology, 11(2), pp. 87-108, 1998. Preliminary version presented at CRYPTO '92.



16. M. O. Rabin, *How to exchange secrets by oblivious transfer*, Technical Report TR-81, Harvard University, 1981.
17. M. O. Rabin, *Transaction Protection by Beacons*, JCSS 27(2), pp. 256-267, 1983.
18. P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Computing, 26(5), pp. 1484-1509, 1997.
19. D. R. Stinson, *On the connections between universal hashing, combinatorial designs and error-correcting codes*, Congressus Numerantium 114, pp. 7-27, 1996.
20. D. R. Stinson *Universal hash families and the leftover hash lemma, and applications to crypto graphy and computing*, 2002. preprint, see, <http://cacr.math.uwaterloo.ca/~dstinson/>
21. A. C. Yao, *How to generate and exchange secrets*, In Proc. 27th IEEE Symposium on the Foundations of Computer Science, pp. 162-167, 1986.

## A Proof of Theorem 2

We extend the proof in Ding [10] to deal with  $OT_1^N$ . We use the same definitions and lemmas as given in [10].

**Definition 2.** Define  $\mathcal{K} \stackrel{\text{def}}{=} \{I \subset \{1, \dots, M\} : |I| = k\}$ .

**Definition 3.** Let  $E \subset GF(2)^M$  and  $I \in \mathcal{K}$ . We say that  $I$  is good for  $E$  if

$$\left| \frac{|\{\alpha \in E : \bigoplus_{i=1}^k \alpha[I[i]] = 0\}|}{|E|} - \frac{|\{\alpha \in E : \bigoplus_{i=1}^k \alpha[I[i]] = 1\}|}{|E|} \right| < 2^{-k/3}.$$

**Definition 4.** Let  $E \subset GF(2)^M$ . We say that  $E$  is fat if  $|E| \geq 2^{0.813M}$ .

**Lemma 8.** [10] For any function  $f : GF(2)^M \rightarrow GF(2)^{M/6}$  and  $\alpha \xleftarrow{R} GF(2)^M$ ,

$$\Pr[f^{-1}(f(\alpha)) \text{ is fat}] > 1 - 2^{-0.02M}.$$

**Definition 5.** For  $\mathcal{A} \subset \{1, \dots, M\}$ , define  $\mathcal{K}_{\mathcal{A}} \stackrel{\text{def}}{=} \{I \subset \mathcal{A} : |I| = k\}$ .

**Definition 6.** For  $\mathcal{A} \subset \{1, \dots, M\}$  and  $E \subset GF(2)^M$ , define

$$\mathcal{B}_E^{\mathcal{A}} \stackrel{\text{def}}{=} \{I \subset \mathcal{K}_{\mathcal{A}} : I \text{ is not good for } E\}.$$

**Lemma 9.** [10] Let  $E \subset GF(2)^M$  be fat. For a uniformly random  $\mathcal{A} \subset \{1, \dots, M\}$  with  $|\mathcal{A}| = n$ ,

$$\Pr \left[ |\mathcal{B}_E^{\mathcal{A}}| < |\mathcal{K}_{\mathcal{A}}| \cdot 2^{-k/6} = \binom{n}{k} \cdot 2^{-k/6} \right] > 1 - 2^{-k/6}.$$

*Proof of Theorem 2:* In order to show the equation (6) of Theorem 2, it suffices to show that with probability  $1 - 2^{-O(k)} - N \cdot 2^{-0.02M}$ , there exists  $\rho \in \{0, 1, \dots, N-1\}$  such that for any  $i \in \{1, \dots, N\}$  and for any distinguisher  $\mathcal{D}$ ,

$$|\Pr[\mathcal{D}(\eta_{N-1}, \Omega, Y_{\rho \oplus i}, Y_\rho) = 1] - \Pr[\mathcal{D}(\eta_{N-1}, \Omega, Y_{\rho \oplus i}, Y_\rho \oplus 1) = 1]| < 2^{-k/3}. \quad (7)$$

Here  $\eta_0 = A_0(\alpha_0), \eta_1 = A_1(\eta_0, \alpha_1), \dots, \eta_{N-1} = A_{N-1}(\eta_{N-2}, \alpha_{N-1})$ ,  $\Omega$  denotes all the information Bob obtains in Step 3-Step 6, and  $Y_0, \dots, Y_{N-1}$  are defined in Step 7 of the protocol.

Note that as in the proof of Theorem 1 in [10] it suffices to show the equation (7) in the case that Bob's recording functions  $A_0, \dots, A_{N-1}$  are deterministic.

We prove a slightly stronger result that the equation (7) hold even if Bob stores not only  $\eta_{N-1}$ , but also  $\eta_0, \eta_1, \dots, \eta_{N-2}$ . Let

$$\begin{aligned} E_0 &\stackrel{\text{def}}{=} \{\alpha \in GF(2)^M : A_0(\alpha) = \eta_0\}, \quad E_1 \stackrel{\text{def}}{=} \{\alpha \in GF(2)^M : A_1(\eta_0, \alpha) = \eta_1\}, \\ \dots, \quad E_{N-1} &\stackrel{\text{def}}{=} \{\alpha \in GF(2)^M : A_{N-1}(\eta_{N-2}, \alpha) = \eta_{N-1}\}. \end{aligned}$$

After  $\eta_0, \dots, \eta_{N-1}$  are computed in Step 2 of the protocol, Bob can compute  $E_0, \dots, E_{N-1}$  using unlimited computing power. But given  $\eta_0, \dots, \eta_{N-1}$ , all Bob knows about  $(\alpha_0, \dots, \alpha_{N-1})$  are that it is uniformly random in  $E_0 \times \dots \times E_{N-1}$ . By Lemma 8, for any recording functions  $A_0, \dots, A_{N-1}$  and for  $\alpha_0, \dots, \alpha_{N-1} \xleftarrow{R} GF(2)^M$ ,

$$\Pr[\text{All of } E_0, \dots, E_{N-1} \text{ are fat}] > 1 - N \cdot 2^{-0.02M} \quad (8)$$

Thus, consider the case that all of  $E_0, \dots, E_{N-1}$  are fat.

Let  $\mathcal{A}^{(0)}, \dots, \mathcal{A}^{(N-1)}$  be the random subsets of  $\{1, \dots, M\}$  with  $|\mathcal{A}^{(i)}| = n, \forall i \in \{0, 1, \dots, N-1\}$ , which Alice chooses in Step 1 of the protocol. By (8) and Lemma 9, we have that for any  $i \in \{1, \dots, N-1\}$ , for  $\rho \in \{0, \dots, N-1\}$ , with probability at least  $1 - N \cdot 2^{-0.02M} - 2^{-k/6+1}$ ,

$$|\mathcal{B}_{E_\rho}^{A^{(\rho)}}|, |\mathcal{B}_{E_{\rho \oplus i}}^{A^{(\rho \oplus i)}}| < \binom{n}{k} \cdot 2^{-k/6}. \quad (9)$$

Thus consider the case that  $\mathcal{B}_{E_\rho}^{A^{(\rho)}}, \mathcal{B}_{E_{\rho \oplus i}}^{A^{(\rho \oplus i)}}$  satisfy (9).

For each  $\epsilon \in \{0, \dots, N-1\}$ , denote  $\mathcal{A}^{(\epsilon)} = \{a_1^{(\epsilon)}, \dots, a_n^{(\epsilon)}\}$ . For  $J = \{j_1, \dots, j_k\} \subset \{1, \dots, n\}$ , denote  $\mathcal{A}_J^{(\epsilon)} = \{a_{j_1}^{(\epsilon)}, \dots, a_{j_k}^{(\epsilon)}\}$ . By Definition 5,  $\mathcal{A}_J^{(\epsilon)} \in \mathcal{K}_{\mathcal{A}^{(\epsilon)}}$ . Define

$$\begin{aligned} F_\rho &\stackrel{\text{def}}{=} \{J \subset \{1, \dots, n\} : |J| = k \wedge \mathcal{A}_J^{(\rho)} \in \mathcal{B}_{E_\rho}^{A^{(\rho)}}\}; \\ F_{\rho \oplus i} &\stackrel{\text{def}}{=} \{J \subset \{1, \dots, n\} : |J| = k \wedge \mathcal{A}_J^{(\rho \oplus i)} \in \mathcal{B}_{E_{\rho \oplus i}}^{A^{(\rho \oplus i)}}\}. \end{aligned}$$

Using (9) and  $|F_\rho| = |\mathcal{B}_{E_\rho}^{A^{(\rho)}}|$ ,  $|F_{\rho \oplus i}| = |\mathcal{B}_{E_{\rho \oplus i}}^{A^{(\rho \oplus i)}}|$ , we have

$$|F_\rho|, |F_{\rho \oplus i}| < \binom{n}{k} \cdot 2^{-k/6}. \quad (10)$$

Consider  $I_0, \dots, I_{N-1}$  defined in Step 5 of the protocol. Let  $\gamma$  be the first  $u$ -bit which Bob sends to Alice in Step 6 of the protocol. Then by (8), (9), (10) and Corollary 1 on the extended interactive hashing, we have that for any strategy Bob uses in Step 3 - Step 6, with probability at least  $1 - 2^{-O(k)} - N \cdot 2^{-0.02M}$ ,  $I_{\gamma \oplus \rho} \notin F_\rho \vee I_{\gamma \oplus \rho \oplus i} \notin F_{\rho \oplus i}$ . WLOG, assume  $I_{\gamma \oplus \rho \oplus i} \notin F_{\rho \oplus i}$ . Let  $Y_\rho = \bigoplus_{j=1}^k \alpha_\rho[a_{I_{\gamma \oplus \rho}}^{(\rho)}[j]]$ ,  $Y_{\rho \oplus i} = \bigoplus_{j=1}^k \alpha_{\rho \oplus i}[a_{I_{\gamma \oplus \rho \oplus i}}^{(\rho \oplus i)}[j]]$  as defined in Step 7 of the protocol. Since  $I_{\gamma \oplus \rho \oplus i} \notin F_{\rho \oplus i}$ , by definition  $\mathcal{A}_{I_{\gamma \oplus \rho \oplus i}}^{(\rho \oplus i)} \notin \mathcal{B}_{E_{\rho \oplus i}}^{\mathcal{A}^{(\rho \oplus i)}}$ . By definition 3 of goodness, for  $\alpha_{\rho \oplus i} \xleftarrow{R} E_{\rho \oplus i}$ ,

$$|\Pr[Y_{\rho \oplus i} = 0] - \Pr[Y_{\rho \oplus i} = 1]| < 2^{-k/3}.$$

Since  $(\alpha_\rho, \alpha_{\rho \oplus i}) \xleftarrow{R} E_\rho \times E_{\rho \oplus i}$ ,  $Y_\rho$  and  $Y_{\rho \oplus i}$  are independent. Thus for any  $b \in GF(2)$ ,

$$|\Pr[Y_{\rho \oplus i} = 0 \mid Y_\rho = b] - \Pr[Y_{\rho \oplus i} = 1 \mid Y_\rho = b]| < 2^{-k/3}$$

which proves (7) and the proof is done.

# In How Many Ways Can You Write Rijndael?

Elad Barkan and Eli Biham

Computer Science Department, Technion – Israel Institute of Technology,  
Haifa 32000, Israel,  
{barkan,biham}@cs.technion.ac.il,  
<http://www.cs.technion.ac.il/~biham/>, <http://tx.technion.ac.il/~barkan/>

**Abstract.** In this paper we ask the question what happens if we replace all the constants in Rijndael, including the replacement of the irreducible polynomial, the coefficients of the MixColumn operation, the affine transformation in the S box, etc. We show that such replacements can create new *dual ciphers*, which are *equivalent* to the original in all aspects. We present several such dual ciphers of Rijndael, such as the square of Rijndael, and dual ciphers with the irreducible polynomial replaced by primitive polynomials. We also describe another family of dual ciphers consisting of the logarithms of Rijndael. We then discuss self-dual ciphers, and extend our results to other ciphers.

## 1 Introduction

Recently, the cipher Rijndael [8] was selected as the Advanced Encryption Standard (AES) [17]. This cipher operates over the algebraic Galois field  $GF(2^8)$ . The motivation for this is computational efficiency, as  $GF(2^8)$  elements can be represented by bytes, which can be very efficiently processed by modern computers, unlike bit-level operations that are usually more expensive in computer power. The drawback is that the algebraic structure inherited by the  $GF(2^8)$  operations may be susceptible to algebraic relations, such as the relations in [9,19]. Algebraic structures may be used to develop cryptographic attacks that exploit the algebraic weakness of the cipher. An example for such attacks are interpolation attacks [11]. In attempt to avoid some of these difficulties, other mechanisms are introduced to these ciphers, such as bit level affine transformations.

In this paper we ask the question what happens if we replace all the constants in Rijndael, including the replacement of the irreducible polynomial, the coefficients of the MixColumn operation, the affine transformation in the S box, etc. We show that such replacements can create new *dual ciphers*, which are *equivalent* to the original in all aspects. Although their intermediate values during encryption are different than Rijndael's, we can show that they are *equivalent* to Rijndael. Examples of such ciphers include ciphers with a primitive polynomial (replacing the irreducible polynomial of Rijndael), the cipher *Square of Rijndael* that encrypts the square of the plaintext under the square of the key to the square of the ciphertext, and a cipher with a triangular affine matrix in the S box.

**Definition 1.** Two ciphers  $E$  and  $E'$  are called *Dual Ciphers*, if they are isomorphic, i.e., if there exist invertible transformations  $f(\cdot)$ ,  $g(\cdot)$  and  $h(\cdot)$  such that

$$\forall P, K \quad f(E_K(P)) = E'_{g(K)}(h(P)).$$

Trivial dual ciphers are very easy to find for all ciphers. For example, every cipher is dual to itself with the identity transformations. Also, for any cipher, the addition of non-cryptographic invertible initial and final transformations creates a trivial dual cipher. We are not interested in these kinds of dual ciphers. The interesting question is whether there exists non-trivial dual ciphers of widely used ciphers.

An extension of dual ciphers, are semi-dual ciphers:

**Definition 2.** A cipher  $E'$  is called a *semi-dual cipher* of  $E$ , if there exist transformations  $f(\cdot)$ ,  $g(\cdot)$  and  $h(\cdot)$  such that

$$\forall P, K \quad f(E_K(P)) = E'_{g(K)}(h(P)).$$

where  $f, g$  and  $h$  are not necessarily invertible (and even not necessarily length-preserving).

Semi-dual ciphers potentially reduce the plaintext, the ciphertext, and the key spaces, and thus may allow to develop efficient attacks on their original cipher.

In this context we would like to mention that interpolation attacks [11] exploit the low order of interpolation polynomial of the cipher. However, they do not exploit other algebraic properties of the interpolation polynomial, such as these used in this paper.

**Definition 3.** In this paper we consider ciphers whose all operations are of the following types:

- Operations in  $GF(2^8)$ :
  1. Addition (i.e., XOR:  $f(x, y) = x \oplus y$ ).
  2. XOR with a constant (e.g.,  $f(x) = x \oplus 3F_x$ ).
  3. Multiplication ( $f(x, y) = x \cdot y$ ).
  4. Multiply by a constant (e.g.,  $f(x) = 03_x \cdot x$ ).
  5. Raise to any power (i.e.,  $f(x) = x^c$ , for any integer  $c$ ). This includes the inverse of  $x$ :  $x^{-1}$ .
  6. Any replacement of the order of elements (e.g., taking a vector containing the elements  $[a, b, c, d]$ , and changing the order to  $[d, c, a, b]$ ).
- Non- $GF(2^8)$  operations:
  7. Linear transformations  $L(x) = Ax$ , for any boolean matrix  $A$ .
  8. Any unary operation over elements in  $GF(2^8)$ . (i.e., a look-up table,  $S(x) = \text{LookUpTable}[x]$  or  $F(x) : \{0, 1\}^8 \rightarrow \{0, 1\}^8$ ).

We call these operations  $EGF(2^8)$  operations.

Please note that this notation implies that in item 7, the variable  $x$ , which is an element in  $GF(2^8)$ , is converted to a vector of 8 bits (in  $GF(2)^8$ ) before being multiplied by the matrix  $A$ . The result is converted back to be an element of  $GF(2^8)$ . A common representation of the vector is as the vector of coefficients of the polynomial  $x$ . It should be noted that since XOR with a constant is also allowed in item 2, any affine transformation is included in the operations we consider (i.e.,  $F(x) = Ax \oplus b$ ).

An example of such a cipher is Rijndael (AES) [8]. Many other ciphers are also built from these operations. Some examples of them are: Shark [6], Square [7], Scream [10], Crypton [13], E2 [16] (without the initial and final permutations), Anubis [2], Khazad [3], and Camellia [1] (with different key scheduling and different FL). Our results can also be extended to Safer++ [14].

In this paper we also deal with the special case of self-duality. That is the case where a cipher is a dual of itself. We study this case and show that such ciphers can be attacked faster than exhaustive search. It is interesting to mention that RSA [18] is an example of a self-dual public key cipher. Let  $e$  and  $n$  be the RSA public key, and let  $c = p^e \pmod{n}$  where  $p$  is the plaintext and  $c$  is the ciphertext. Then it follows that RSA is a dual of itself:  $c^2 = (p^2)^e \pmod{n}$ .

We also discuss the family of *Log Dual Ciphers*. In a log dual cipher, the logarithm of the plaintext is encrypted by the logarithm of the key to the logarithm of the ciphertext. We show that Rijndael has a family of log dual ciphers.

We indicate a variety of possible applications for dual ciphers, ranging from gaining insight for differential [4] and linear [15] cryptanalysis, to speeding up encryption, and to protect against power analysis [12] and fault analysis [5].

This paper is organized as follows: In section 2 we give a short description of Rijndael. Section 3 shows how to define square dual ciphers. Section 4 deals with changing the irreducible polynomial. Section 5 shows how to define logarithmic dual ciphers. In section 6 we discuss the special case of self-duality and show how to mount an attack on self-dual ciphers. Section 7 deals with application to other ciphers. Section 8 deals with other applications of dual ciphers. The paper is summarized in Section 9.

## 2 Description of Rijndael

In this section we give a short description of Rijndael. For a full description of Rijndael the reader may consult [8,17]. Rijndael is a block cipher with 128-bit blocks, and three key sizes of 128, 192 and 256 bits. The 128-bit blocks are viewed as either 16 bytes or as four 32-bit words. The bytes are organized in a square form:

$b_0$	$b_4$	$b_8$	$b_{12}$
$b_1$	$b_5$	$b_9$	$b_{13}$
$b_2$	$b_6$	$b_{10}$	$b_{14}$
$b_3$	$b_7$	$b_{11}$	$b_{15}$

where  $b_i$  notes the  $i$ 'th byte of the block.

Each column in this representation can be viewed as a 4-byte word. Rijndael has operations that work on columns, operations that work on rows, and operations that work on each byte separately.

The plaintexts are encrypted through successive operation of a round function 10 times (or 12 or 14 times for 192-bit and 256-bit keys, respectively).

A round is composed of 4 consecutive operations:

1. ByteSub: An S box is applied to each byte of the data (16 times in parallel).
2. ShiftRow: Changing the order of bytes in the data.
3. MixColumn: Every 4 consecutive bytes (column) are mixed by a linear operation.
4. AddRoundKey: The data is XORed with a 128-bit subkey.

The S box of Rijndael is taking the multiplicative inverse of the input in  $GF(2^8)$  (modulo the irreducible polynomial of Rijndael  $x^8 + x^4 + x^3 + x + 1$ , which is denoted in binary notation by  $11B_x$ ; for the purpose of inversion the inverse of  $00_x$  is defined to be  $00_x$ ), the output of which is transformed by the affine transformation:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

where the  $x_i$ 's and the  $y_i$ 's are coefficients of  $x$  and  $y$  (i.e., the bits of the bytes), and  $x_0$  and  $y_0$  are the least significant bits.

The ShiftRow operation is defined as changing the order of bytes in the data. When viewing the data in its square form, ShiftRow is:

- Leaving the first first row unchanged.
- Shifting the second row by one byte to the left (cyclically).
- Shifting the third row by two bytes to the left (cyclically).
- Shifting the fourth row by three bytes to the left (cyclically).

Taking the square form as the input of the ShiftRow operation, the ShiftRow operation has the following effect:

$$\begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_1 & b_5 & b_9 & b_{13} \\ \hline b_2 & b_6 & b_{10} & b_{14} \\ \hline b_3 & b_7 & b_{11} & b_{15} \\ \hline \end{array} \xrightarrow{\text{ShiftRow}} \begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_5 & b_9 & b_{13} & b_1 \\ \hline b_{10} & b_{14} & b_2 & b_6 \\ \hline b_{15} & b_3 & b_7 & b_{11} \\ \hline \end{array}$$

The MixColumn operation mixes every 4 consecutive bytes (every column) of the data. Therefore, there are 4 mix operations in each round. Let  $b_i$ ,  $b_{i+1}$ ,

$b_{i+2}$ ,  $b_{i+3}$  be consecutive bytes of a column. The new state is defined by

$$\begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_1 & b_5 & b_9 & b_{13} \\ \hline b_2 & b_6 & b_{10} & b_{14} \\ \hline b_3 & b_7 & b_{11} & b_{15} \\ \hline \end{array} \xrightarrow{\text{MixColumn}} \begin{array}{|c|c|c|c|} \hline b'_0 & b'_4 & b'_8 & b'_{12} \\ \hline b'_1 & b'_5 & b'_9 & b'_{13} \\ \hline b'_2 & b'_6 & b'_{10} & b'_{14} \\ \hline b'_3 & b'_7 & b'_{11} & b'_{15} \\ \hline \end{array}$$

where  $i \in \{0, 4, 8, 12\}$ , and all the operations are in  $GF(2^8)$ :

$$\begin{pmatrix} b'_i \\ b'_{i+1} \\ b'_{i+2} \\ b'_{i+3} \end{pmatrix} = \begin{pmatrix} 02_x & 03_x & 01_x & 01_x \\ 01_x & 02_x & 03_x & 01_x \\ 01_x & 01_x & 02_x & 03_x \\ 03_x & 01_x & 01_x & 02_x \end{pmatrix} \begin{pmatrix} b_i \\ b_{i+1} \\ b_{i+2} \\ b_{i+3} \end{pmatrix}$$

This operation is actually a multiplication of the column by the polynomial  $c(x) = 03_x x^3 + 01_x x^2 + 01_x x + 02_x$  in  $GF(2^8)^4$  modulo the polynomial  $x^4 + 1$ .

The AddRoundKey simply XORs the 128-bit subkey to the data. The subkey is generated by the key expansion.

The key expansion of Rijndael generates the subkeys from the key using a blend of the same operations used in the rest of Rijndael, and using the round constants  $Rcon[i] = (02_x)^{i-1}$  ( $i$  starts at 1).

The round-function of the first and the last rounds are slightly different than in other rounds: In the first round there is an additional AddRoundKey operation before the round starts, and in the last round the MixColumn operation is eliminated.

When the key size is 128 bits the round-function is repeated 10 times. The number of rounds is higher when longer keys are used: there are 12 rounds when the key size is 192 bits, and 14 rounds when the key size is 256 bits.

### 3 Square Dual Ciphers

Given a cipher  $E$  that uses only operations of  $EGF(2^8)$ , we define the cipher  $E^2$  by modifying the constants of  $E$ . All the operations that do not involve constants remain unchanged. There are only four operations that involve constants:

1.  $f(x) = c \cdot x$ .
2.  $f(x) = c \oplus x$ .
3.  $L(x) = Ax$ , where  $A$  is a constant matrix.
4.  $S(x) = LookUpTable[x]$ , where the look-up table is constant.

In the first two operations we change the constant  $c$  in  $E$  to be  $c^2$  in  $E^2$ , where  $c^2$  is the result of squaring  $c$  in  $GF(2^8)$ . In the affine transformation  $A$  is replaced by  $QAQ^{-1}$ , where in the case of Rijndael  $Q$  and  $Q^{-1}$  are:



$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad Q^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad (1)$$

From now on we denote  $QAQ^{-1}$  by  $A^2$ , as we will show later that for any  $x$ ,  $QAQ^{-1}x^2 = QAx = (Ax)^2$ .  $A^2$  of Rijndael is given in Appendix A. The matrices  $Q$  and  $Q^{-1}$  depend on the irreducible polynomial of  $GF(2^8)$ . The matrices above suit Rijndael's irreducible polynomial  $x^8 + x^4 + x^3 + x + 1$ .

Finally, we replace look-up tables of the form  $S(x)$  with  $S^2(x)$ , where  $S^2(x)$  is defined as  $S^2(x) = QS(Q^{-1}x)$ .

**Remark:** To make it clear, in our notation,  $E^2$  is not  $E(E(\cdot))$  nor  $(E(\cdot))^2$ ,  $A^2$  is not the matrix  $A$  multiplied with itself, and  $S^2(x)$  is not  $(S(x))^2$ , nor  $S(S(x))$ .

In general terms, to specify  $E^2$ , we take the specifications of the cipher, raise all the constants in the cipher to their second power, replace matrices  $A$  by  $A^2 = QAQ^{-1}$  and replace look-up tables  $S(x)$  by  $S^2(x) = QS(Q^{-1}x)$ . If we take Rijndael as an example of  $E$ , the polynomial  $03_x x^3 + 01_x x^2 + 01_x x + 02_x$  of the mix column operation is replaced by  $05_x x^3 + 01_x x^2 + 01_x x + 04_x$ .<sup>1</sup> As a result of the above replacements, the affine transformation  $Ax + b$  is replaced by the affine transformation  $A^2x + b^2 = QAQ^{-1}x + b^2$ .

The key expansion consists of S boxes, XORs, and XORs with constants in  $GF(2^8)$  (called *Rcon*) which are powers of  $02_x$ . These operations are replaced by the replacement operations as mentioned above, with the *Rcon* constants being replaced by their squares.

We will now show that  $E$  and  $E^2$  are dual ciphers:

**Theorem 1.** For any  $K$  and  $P$ ,  $E^2_{K^2}(P^2) = (E_K(P))^2$ .

In the context of this paper, the notation  $K^2$ , and  $P^2$  denote the square operation of each byte of  $K$  and  $P$  (and similarly for any data block).

This theorem states that if  $P$  is the plaintext,  $K$  is the key and the result of encryption with cipher  $E$  is  $C$ , then the result of encrypting  $P^2$  under the key  $K^2$  with the cipher  $E^2$  is necessarily  $C^2$ .

**Proof.** Any Galois field is congruent to a Galois field of the form of  $GF(q^m)$ , where  $q$  is a prime. The number  $q$  is called the characteristic of the field. It is well known that for any  $a, b \in GF(q^m)$  it follows that:  $(a + b)^q = a^q + b^q$ . In  $GF(2^8)$ :  $(a + b)^2 = a^2 + b^2$ . That actually means that squaring an element in  $GF(2^8)$  is a linear operation, which can be applied by a multiplication by a binary matrix  $Q$  of size  $8 \times 8$ . We computed the matrix  $Q$  of Rijndael, and described it in Eq. (II). It follows that  $Q^{-1}$  is the matrix that takes out the square root of an element in  $GF(2^8)$ .

<sup>1</sup> In  $GF(2^8)$ ,  $03_x^2 = 05_x$ .

To complete the proof, it suffices to show that for each operation  $f(x)$  in  $E$ , and the corresponding operation in  $E^2$ , which we denote in this proof by  $f^2(x)$ , it follows that  $f^2(x^2) = (f(x))^2$ :

1.  $f(x, y) = x \oplus y$ . In this case  $f^2(x^2, y^2) = x^2 \oplus y^2 = (x \oplus y)^2 = (f(x, y))^2$ .
2.  $f(x) = x \oplus c$ . By definition  $f^2(x^2) = x^2 \oplus c^2 = (x \oplus c)^2 = (f(x))^2$ .
3.  $f(x, y) = x \cdot y$ . In this case  $f^2(x^2, y^2) = x^2 \cdot y^2 = (x \cdot y)^2 = (f(x, y))^2$ .
4.  $f(x) = x \cdot c$ . By definition  $f^2(x^2) = x^2 \cdot c^2 = (x \cdot c)^2 = (f(x))^2$ .
5.  $f(x) = x^c$ . In this case  $f^2(x^2) = (x^2)^c = (x^c)^2 = (f(x))^2$ .
6. It is clear that replacing the order of elements after they are raised to their second power is equal to raising elements to their second power, and then replacing their order.
7.  $f(x) = L(x) = Ax$ . By definition  $f^2(x^2) = L^2(x^2) = QAQ^{-1}x^2 = QAx = (Ax)^2 = (f(x))^2$ , as  $Q$  is the matrix which corresponds to the squaring operation in  $GF(2^8)$ .
8.  $f(x) = S(x) = \text{LookupTable}[x]$ . By definition

$$f^2(x^2) = S^2(x^2) = QS(Q^{-1}x^2) = QS(x) = (S(x))^2 = (f(x))^2.$$

■

The cipher  $E^4 = (E^2)^2$  is a dual cipher of  $E^2$ , and thus also of  $E$ . Moreover, all ciphers  $E^{2^i}$  (for all  $i$ ), which are  $E, E^2, E^4, E^8, E^{16}, E^{32}, E^{64}$  and  $E^{128}$ , are all dual ciphers of each other (there are 8 such ciphers as  $E^{2^8} = E$ ).

It is interesting to note that Rijndael have these 7 dual ciphers, independently of the key size, the block size, the number of rounds, and even the order of operations in the cipher. These dual ciphers exist for any cipher whose all operations are  $EGF(2^8)$  operations.

Note that it is possible to define a trivial square dual cipher for any cipher by taking a cipher  $E$  and defining  $E^2$  which apply  $Q^{-1}$  on the plaintext and  $K$ , calls  $E$ , and then applies  $Q$  on the result. However, we are interested in non-trivial dual ciphers with different cores.

## 4 Modifying the Polynomial

An  $EGF(2^8)$  cipher  $E$  can include multiplication modulo an irreducible polynomial. The irreducible polynomial in Rijndael is used for the inverse computation in the S box and also in the multiplications in the MixColumn operation. Several researchers asked why the irreducible polynomial of Rijndael was not selected to be primitive. There are 30 irreducible polynomials of degree 8, of which 16 are primitive. In our discussion it is irrelevant if the irreducible polynomial is primitive or not, due to the isomorphism of all fields of  $GF(2^8)$ . The isomorphism transformation that takes one description of a cipher under an irreducible polynomial  $g(x)$  to another description with a different irreducible polynomial  $\hat{g}(x)$  is linear, and therefore can be represented as a binary matrix  $R$  such that  $y = R \cdot x$ , where  $x$  is the vector representation of an element under Rijndael's

$g(x)$  polynomial, and  $y$  is the representation under the new polynomial  $\hat{g}(x)$ . The matrix  $R$  is used in a similar way to the matrix  $Q$  of the square dual cipher. The change of operations in the cipher is also similar to the square dual cipher. Note that the  $x^2$  operation there is equivalent to  $Q \cdot x$ , and the same proof of duality follows.

The  $R$  matrix is always of the form  $R = (1, a, a^2, a^3, a^4, a^5, a^6, a^7)$ , where the  $a^i$ 's are computed modulo the irreducible polynomial  $\hat{g}(x)$ . Note that the matrix  $Q$  is actually one of these matrices  $R$ . For each irreducible polynomial we can define its 8 square dual ciphers. Since there are 30 irreducible polynomials, we get that there are 240 dual ciphers for each  $EGF(2^8)$  cipher.

For example, we describe one of these 240 dual ciphers of Rijndael: the irreducible polynomial of Rijndael is replaced by the primitive polynomial  $x^8 + x^4 + x^3 + x^2 + 1$  (denoted in binary notation by  $11D_x$ ). In this example, the  $R$  matrix is

$$R = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad R^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The inverse matrix  $R^{-1}$  takes an element of the dual cipher to Rijndael's representation. It is interesting to note that the affine matrix of the S box becomes lower triangular in this case:

$$\hat{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Also, the constant  $63_x$  in the S box becomes  $64_x$ , and the coefficients  $03_x, 02_x$  of the MixColumn operation are interchanged (i.e., to  $02_x, 03_x$ ). The coefficients  $0B_x, 0D_x, 09_x, 0E_x$  are also interchanged in pairs to  $0D_x, 0B_x, 0E_x, 09_x$ . The  $Rcon$  constants  $(02_x)^{i-1}$  are replaced by  $(03_x)^{i-1}$ .

Thus, we conclude that the choice of the irreducible polynomial of Rijndael is arbitrary, and in particular, there is no advantage to selecting a primitive polynomial over the current polynomial of Rijndael.

## 5 Log Dual Ciphers

In this section we discuss dual ciphers, to which we call *log dual ciphers*. We actually describe a family of log dual ciphers, which differ slightly from each other.

Let  $g$  be a generator in  $GF(2^8)$ . Since the cipher works on elements of  $GF(2^8)$  we can write any element  $x$  as an exponent of  $g$ , i.e.,  $x = g^i$ , except for  $x = 0$ , which we define as  $g^{-\infty}$ . In a logarithmic notation we write:  $\log_g x = i$ , where  $\log_g 0 = -\infty$ . In the log cipher we use the logarithm representation of the elements, instead of the polynomial representation used in the original description of the cipher.

Let  $x$  and  $y$  be elements of  $GF(2^8)$ , and let  $i = \log_g x$ ,  $j = \log_g y$ .

We use the notation  $E^{\log_g}$ , or shortly  $E^{\log}$ , to denote the log dual cipher. The log dual cipher is defined by taking the specifications of the cipher, and replacing the following operations:

1. The operation  $f(x, y) = x \oplus y$  is replaced by the operation  $f^{\log}(i, j) = j + T(i - j) \pmod{255}$  or by  $f^{\log}(i, j) = i + T(j - i) \pmod{255}$ , where  $T(i)$  is defined as  $T(i) = \log_g(g^i \oplus 1)$ . In cases where  $-\infty$  appears in  $f^{\log}$ , we define  $f^{\log}(-\infty, j) = j$ , and  $f^{\log}(i, -\infty) = i$ .
2. The operation  $f(x) = x \oplus c$  is replaced by the operation  $f^{\log}(i) = k + T(k - i) \pmod{255}$  where  $k = \log_g c$ .
3. The operation  $f(x, y) = x \cdot y$  is replaced by the operation  $f^{\log}(i, j) = i + j \pmod{255}$ . If either  $x$  or  $y$  is  $-\infty$ , then the result is  $-\infty$ .
4. The operation  $f(x) = x \cdot c$  is replaced by the operation  $f^{\log}(i) = i + k \pmod{255}$ , where  $k = \log_g c$ .
5. The operation  $f(x) = x^m$  is replaced by the operation  $f^{\log}(i) = i \cdot m \pmod{255}$ . If  $i = -\infty$  then the result is  $-\infty$ .
6. Replacement of the order of elements remains the same replacement of order of the elements.
7. The operation  $S(x) = \text{LookupTable}[x]$  is replaced by the operation  $S^{\log}(i) = \log_g(S(g^i))$ .
8. The linear transformation  $L(x) = Ax$  is treated as a lookup table (like in the previous item).

The definition of  $\log_x(0) = -\infty$  is made carefully, ensuring that this definition is consistent: When applying the operation  $j + T(i - j)$ , a  $-(-\infty)$  might result as an argument to  $T$ . We define that  $j + (-\infty) = -\infty$  (which corresponds to multiplication by 0 in the original cipher or to XOR of a value with itself),  $-\infty \cdot c = -\infty$  (which corresponds to an exponentiation of 0 in the original cipher),  $-\infty - (-\infty) = 0$  (which corresponds to  $0 \oplus 0$ , or to  $x \oplus 0$ ),  $i - (-\infty) \neq j - (-\infty)$  for  $i \neq j$  (meaning that  $-(-\infty)$  does not consume  $i$ ).  $T(-\infty) = 0$ ,  $T(0) = -\infty$ .  $T(i - (-\infty)) = -(-\infty) + i$ . Note that the  $-(-\infty)$  is always a result of an application of  $T$ . Then, another  $+(-\infty)$  is always waiting to cancel it (as  $j$ ). Therefore, the result of the  $T$  operation is always a number or a  $-\infty$ .

The following theorem proposes that if  $P$  is the plaintext,  $K$  is the key and the result of encryption of  $P$  under the key  $K$  with cipher  $E$  is  $C$ , the result

of encrypting  $\log_g(P)$  under the key  $\log_g(K)$  with the cipher  $E^{\log}$  is necessarily  $\log_g(C)$ .

**Theorem 2.** *Let  $g$  be a generator in  $GF(2^8)$ . For any  $K$  and  $P$ :*

$$E_{\log_g K}^{\log}(\log_g P) = \log_g(E_K(P)).$$

In the context of this paper  $\log_g X$  denotes the log of each byte of  $X$ .

**Proof.** It suffices to show that for each operation  $f(x)$  in  $E$ , and the corresponding operation in  $E^{\log}$ , which we denote by  $f^{\log}(x)$ , it follows that  $f^{\log}(\log_g x) = \log_g(f(x))$ .

1.  $f(x, y) = x \oplus y$ . By definition  $f^{\log}(i, j) = j + T(i - j) = j + \log_g(g^{i-j} \oplus 1) = \log_g(g^j \cdot (g^{i-j} \oplus 1)) = \log_g(g^i \oplus g^j) = \log_g(x \oplus y) = \log_g(f(x, y))$ .
2.  $f(x) = x \oplus c$ , in the same way as the previous item.
3.  $f(x, y) = x \cdot y$ . In this case  $f^{\log}(i, j) = i + j = \log_g(g^{i+j}) = \log_g(x \cdot y) = \log_g(f(x, y))$ .
4.  $f(x) = x \cdot c$ , in the same way as the previous item.
5.  $f(x) = x^c$ . In this case  $f^{\log}(i) = i \cdot c = \log_g(x^c) = \log_g(f(x))$ .
6. It is clear that replacing the order of elements after their log-value is taken is equal to replacing the order of elements and then taking their log-value.
7.  $f(x) = S(x) = \text{LookUpTable}[x]$ . In this case, by definition of  $f^{\log}$  it follows that:  $f^{\log}(i) = S^{\log}(i) = \log_g(S(g^i)) = \log_g(S(x)) = \log_g(f(x))$ .
8.  $L(x) = Ax$  is considered like a table in the previous item, and is treated in the same way.

The above equations hold also in the case that  $-\infty$  is an argument. ■

Note that the non-linear part of the ByteSub transformation of Rijndael in the log dual cipher becomes very simple (and linear). The non-linear part is finding the multiplicative inverse of an element. This operation is replaced by negation in the log dual cipher:

$$x^{-1} \longrightarrow -i.$$

The  $T$  transformation is non-linear. It has interesting properties. Here are some of the properties of the  $T$  transformation:

1.  $T(x) - T(-x) = x$
2.  $T(2x) = 2T(x)$  (therefore,  $\forall i, T(2^i x) = 2^i T(x)$ )
3.  $T(T(x)) = x$
4. Let  $g \triangleq g'^y, yT_g(x) = T_{g'}(yx)$
5.  $T_g = T_{g^{2^i}}$
6.  $T(x) = -T(-T(-x))$
7.  $T(85) = 170, T(170) = 85$ , and if  $T(x) = x/2$  then  $x \in \{85, 170\}$ . Note that  $85/2 \equiv 170 \pmod{255}$
8.  $T(0) = -\infty$ .

**Table 1.** The Table  $T(x)$  with the generator  $03_x$  with the irreducible polynomial  $11B_x$  (Rijndael).

$+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	$-\infty$	25	50	223	100	138	191	112	200	120	21	245	127	99	224	33
16	145	68	240	92	42	10	235	196	254	1	198	104	193	181	66	45
32	35	15	136	32	225	179	184	106	84	157	20	121	215	31	137	101
48	253	197	2	238	141	147	208	63	131	83	107	82	132	186	90	55
64	70	162	30	216	17	130	64	109	195	236	103	199	113	228	212	174
80	168	160	59	57	40	170	242	167	175	203	62	209	19	158	202	176
96	251	190	139	13	4	47	221	74	27	248	39	58	161	71	126	246
$T[x] =$ 112	7	76	166	243	214	122	164	153	9	43	117	183	180	194	110	12
128	140	239	69	56	60	250	177	144	34	46	5	98	128	52	218	150
144	135	16	217	53	206	188	143	178	226	119	201	159	169	41	93	155
160	81	108	65	182	118	227	114	87	80	156	85	211	229	232	79	88
176	95	134	151	37	124	29	163	123	38	249	61	204	149	219	97	6
192	247	28	125	72	23	49	26	75	8	154	94	89	187	207	148	205
208	54	91	241	171	78	233	116	44	67	146	142	189	252	102	237	3
224	14	36	152	165	77	172	231	230	173	213	244	22	73	222	51	129
240	18	210	86	115	234	11	111	192	105	185	133	96	220	48	24	—

and

$$T[-\infty] = 0.$$
$$T[i - (-\infty)] = -(-\infty) + i$$

- 9.  $T(-\infty) = 0$ .
- 10.  $T(x) = T(x \pm 255)$  - The cycle size of  $T$  is 255.

Proofs of the properties of  $T(x)$  will appear in the full paper.

The table of  $T(x)$  with the generator  $03_x$  is described in Table [11](#).

Each one of the 240 mentioned representations of Rijndael has the same set of 128 log dual ciphers.

## 6 Self-Dual Ciphers

We mention that any cipher is trivially dual to itself. However, it is possible to find ciphers that are self-dual in a non-trivial way. One such interesting family of dual ciphers is square dual ciphers. Let  $E$  be a square self-dual cipher. It follows that:

$$(E_K(P))^2 = E_{K^2}(P^2).$$

This means that each constant is the square of itself. In  $GF(2^8)$  it means that the constants are either 0 or 1.

If we take Rijndael as an example, we need to change the constant  $63_x$  in the affine transformation in the S box to either  $00_x$  or  $01_x$ . We would also need to

change the constants of the mix column operation. A possible alternative matrix for the mix column operation, whose entries consist of only 0's and 1's is:

$$M = M^{-1} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

In the key expansion we need to change the round constant. Any selection of values from  $\{0_x, 1_x\}$  can be made for the *Rcon* constants. There are various such selections that can still prevent related key attacks.

We can replace the affine transformation to a self-dual one. We can easily find 8 affine transformations that are self-squares: The matrix  $Q$  (shown in Eq. (11)) is the square of itself under our definition, since  $Q^2 = Q(Q)Q^{-1} = Q$ . The order of  $Q$  is 8, therefore, we can easily find 8 self-square affine transformations:  $Q$ ,  $Q \cdot Q$ ,  $Q \cdot Q \cdot Q$ ,  $\dots$ ,  $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q$  and  $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q = I$ . Notice that all the linear combinations with coefficients from  $\{0_x, 1_x\}$  of these matrices, are also self-squares matrices. Therefore, there are 256 such self-square matrices. Detailed analysis shows that these are all the self-square matrices. Of these 256 matrices only 128 matrices are involutions.

### 6.1 Higher Order Self-Dual Cipher

In Section 6 we introduced the square self-dual cipher. In a similar way we can define the 4'th power self-dual cipher. Let  $E$  be a 4'th power self-dual cipher. It follows that:

$$(E_K(P))^4 = E_{K^4}(P^4).$$

This means that each constant is the 4'th power of itself. There are 4 such elements: the elements 0 and 1, and the two elements of order 3 (which are  $g^{85}$ ,  $g^{170}$ , where  $g$  is a generator).

We need the affine transformation to be self-dual, and therefore:  $A^4 = Q \cdot Q \cdot (A) \cdot Q^{-1} \cdot Q^{-1} = A$ . We can see that  $Q$ ,  $Q \cdot Q$ ,  $Q \cdot Q \cdot Q$ ,  $\dots$ ,  $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q$  and  $Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q \cdot Q = I$  solve it much like for the square self-dual cipher, along with all their linear combinations, with coefficients from  $\{0_x, 1_x, g^{85}, g^{170}\}$  (which is  $GF(2^2)$ ). The total number of linear combinations is  $2^{16}$ , of which  $3 \cdot 2^{13}$  are involutions.

For the 16'th power square self-dual cipher all the constants should be 0, 1, and all the 14 elements of orders 3, 5, and 15. The 16'th power square self-dual matrices are all the linear combinations of the  $Q^i$  matrices, with coefficients from the above constants. The total number of 16'th power square self-dual matrices is  $2^{32}$ , of which  $7 \cdot 5 \cdot 3^2 \cdot 2^{22}$  matrices are involutions. Fortunately, Rijndael's matrix is none of these matrices.

## 6.2 Cryptanalysis of Self-Dual Ciphers

The self-dual property of a cipher can be used to mount an attack which reduces the complexity of exhaustive search by a factor of about 8 in the case above (or by a factor of the number of the self-duals in the more general case). For example, if the key size is 128 bit, exhaustive search takes  $2^{128}$  operations, and the attack we propose requires about  $2^{125}$  operations. If we consider the expected time to complete the attack, exhaustive search takes about  $2^{127}$ , and our attack takes about  $2^{124}$  operations.

It is interesting to note that the number of rounds of the cipher does not affect the complexity of this attack.

By using the following chosen plaintext attack the key can be discovered in  $2^{125}$  operations using 8 chosen plaintexts.

The attack takes advantage of cycles of keys under the squaring operation: A cycle is a set of keys where each key is the square of its predecessor, i.e.,  $\{K', K'^2, \dots, K'^{2^7}\}$ , and where the square of the last element equals the first element:  $K' = K'^{2^8}$ . Note that the possible cycle lengths are 8, 4, 2, and 1.

1. Choose a plaintext  $P$ , and compute  $P_i = P^{2^i}$ , for  $i = 0, \dots, 7$ .
2. Ask for the encryption of  $P_0, \dots, P_7$ , and denote the corresponding ciphertexts by  $C_0, \dots, C_7$ . For every  $i$ , compute  $\hat{C}_i = (C_i)^{2^{-i}}$ , where the square root is defined to be the operation that finds for every byte its square root in  $GF(2^8)$  (there is only one square root for each value).
3. Choose one key  $K'$  in each cycle, and compute  $C = E_{K'}(P_0)$ . If  $C = \hat{C}_i$  for some  $i \in \{0, \dots, 7\}$ ,  $K'^{2^i}$  is a candidate to be  $K$ . Otherwise,  $K$  is not one of  $\{K'^{2^i}\}$ .

An equality  $C = \hat{C}_i$  in step 3 ensures that encryption of  $P_i$  under the key  $K'^{2^i}$  gives  $C_i$ : If  $C = \hat{C}_i$ , then  $C^{2^i} = \hat{C}_i^{2^i} = C_i$ . Therefore,  $C^{2^i} = (E_{K'}(P_0))^{2^i} = C_i = E_K(P_0^{2^i}) = E^{2^i}_K(P_0^{2^i})$ . From the self-duality property it follows that:  $K = K'^{2^i}$  (or that this is a false-alarm).

Note that the correct key is always found by this method, since for the correct key  $K$ :  $E_K(P_0) = C_0$ . The self-duality property implies that this happens if and only if for any  $i$ ,  $E_{K^{2^i}}(P_0^{2^i}) = C_0^{2^i}$ . For each cycle, for example, for  $\{K', K'^2, \dots, K'^{2^7}\}$ , we test only one key. If this key is  $K$ , then we would find it on the first equation. If one of the other keys is  $K$ , then the corresponding equation holds. So by checking one key out of a cycle we cover the whole cycle.

We test about 8 keys for every trial encryption. It is easy to choose the keys  $K$  in such a way that we choose only one key out of each cycle of keys. Therefore, this attack finds the key in about  $2^{125}$  time. In the full version of this paper we analyze the complexity of the attack and show how to enumerate the keys (choose only one key of each cycle), and show that the total number of cycles, and thus, the maximal complexity of this attack, is  $2^{125} + 2^{61} + 2^{30} + 2^{15}$ , using 8 chosen plaintexts. The average case complexity is  $2^{124} + \varepsilon$  where  $\varepsilon = 2^{-4} + 2^{-67} + 2^{-98}$ .

We note that a similar attack can be designed for higher order self-dual ciphers.



## 7 Application to Other Ciphers

Square [7], Scream [10], Anubis [2], Crypton [13] and Khazad [3] are all  $EGF(2^8)$  ciphers. E2 [16] (without the initial and final permutations), and Camellia [1] (with different key scheduling and different FL) are also  $EGF(2^8)$  ciphers. Thus, our results hold to these ciphers as well.

Our work can be extended to include ciphers such as Safer++ [14]. The only operation in Safer++ that is not a  $EGF(2^8)$  operation is addition modulo 256:  $f(x, y) = x + y \pmod{256}$ . For the square dual cipher, we can define  $f^2(x^2, y^2) = (f(x, y))^2$ .  $f^2$  can be implemented by  $f^2(x, y) = Qf(Q^{-1}x, Q^{-1}y)$ . This results in a substitution table of size  $2^{16}$ .

It should be noted that since Safer++ does not use  $GF(2^8)$  multiplications or exponentiations, the irreducible polynomial is irrelevant. For such a cipher, we can create a wide range of dual ciphers by using any invertible binary matrix  $Q$  of size  $8 \times 8$ . The operation  $f^Q(x)$  is defined as  $f^Q(x) = Qf(Q^{-1}x)$ . For operations with two parameters  $f^Q(x, y) = Qf(Q^{-1}x, Q^{-1}y)$ . A constant  $c$  is replaced by  $Qc$ . This change does not fundamentally change the differential [4] properties of such functions, since  $Q$  and  $Q^{-1}$  are linear and invertible.

If we take E2, and remove the initial and final transformations, the affine operation, and also change the  $v_{-1}$  value of the key scheduling to be composed only from 0's and 1's, then E2 is a self-dual cipher. That means that the attack we present in this paper is also applicable to this variant.

## 8 Other Applications

A possible application of dual ciphers is for developing differential [4] or linear [15] attacks. In such cases the insight gained from the dual ciphers can be used to attack the dual cipher, an attack which can be easily transformed to the original. A possible example for such insight might be the simplification of the affine transformation in the S box to a triangular matrix (see Section 4), which reduces the effect of modifying bits in the input on the resultant output of this transformation.

An other interesting application of dual ciphers might be an optimization of the speed of the cipher, as in some cases the dual cipher might actually be faster to compute than the original cipher! For example, many ciphers include multiplications by constants. The Hamming weight and the size of the constant has implications on the implementation efficiency. Thus, finding a more efficient dual cipher might be a good optimization strategy. Also, in some cases encryption might be fastest using one dual cipher, and decryption be fastest using another dual cipher.

The existence of dual ciphers can also be used to protect implementation against fault-analysis [5] and power-analysis [12], by selecting a different dual cipher at random each time an encryption or decryption is desired.

## 9 Summary

In this paper we show how to write many different implementations of Rijndael using its various dual ciphers. We describe hundreds of non-trivial dual ciphers of Rijndael, many of them differ from Rijndael only by the replacement of constants. We also discuss an attack on self-dual ciphers.

We conclude that the irreducible polynomial of Rijndael is chosen arbitrarily, and that it is possible to replace the irreducible polynomial of Rijndael by any other irreducible or primitive polynomial without changing the strength of cipher, and even without changing the cipher itself.

## Acknowledgments

We are pleased to thank Ronny Roth for the various discussions which helped improving the results of this paper and to John Kelsey for observing that dual ciphers may be used to prevent power analysis.

The work described in paper has been supported by the European Commission through the IST Programme under Contract IST-1999-12324.

## References

1. Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moria, Junko Nakajima, Toshio Tokita, *Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis*, submitted to NESSIE, 2000.
2. Paulo S.L.M. Barreto, Vincent Rijmen, *The Anubis Block Cipher*, submitted to NESSIE, 2000.
3. Paulo S.L.M. Barreto, Vincent Rijmen, *The Khazad Legacy-Level Block Cipher*, submitted to NESSIE, 2000.
4. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
5. Eli Biham, Adi Shamir, *Differential Fault of Secret-Key Cryptosystems*, Advances in Cryptology, proceedings of Crypto'97, Lecture Notes in Computer Science 1294, Springer-Verlag, pp. 513–525, 1997.
6. Antoon Bosselaers, Joan Daemen, Erik De Win, Bart Preneel, Vincent Rijmen, *The Cipher Shark*, proceedings of Fast Software Encryption '96, Lecture Notes in Computer Science 1039, Dieter Gollmann, Ed., Springer-Verlag, pp. 99–112, 1996.
7. Joan Daemen, Lars R. Knudsen, Vincent Rijmen, *The Block Cipher Square*, proceedings of Fast Software Encryption '97, Lecture Notes in Computer Science 1267, Eli Biham, Ed., Springer-Verlag, pp. 149–165, 1997.
8. Joan Daemen, Vincent Rijmen, *AES Proposal: Rijndael*, submitted to the Advanced Encryption Standard (AES) contest, 1998.
9. Niels Ferguson, Richard Schroeppel, Doug Whiting, *A Simple Algebraic Representation of Rijndael*, proceedings of Selected Areas in Cryptography, Lecture Notes in Computer Science 2259, Serge Vaudenay and Amr Youssef, Eds., Springer-Verlag, pp. 103–111, 2001.
10. Shai Halevi, Don Coppersmith, Charanjit Jutla, *Scream: a Software-Efficient Stream Cipher*, preproceedings of Fast Software Encryption 2002, pp. 190–204, 2002.

11. Thomas Jakobsen, Lars R. Knudsen , *The Interpolation Attack on Block Ciphers*, proceedings of Fast Software Encryption '97, Lecture Notes in Computer Science 1267, Eli Biham, Ed., Springer-Verlag, pp. 28–40, 1997.
12. Paul Kocher, Joshua Jaffe, Benjamin Jun, *Differential Power Analysis*, Advances in Cryptology, proceedings of Crypto'99, Lecture Notes in Computer Science 1666, Springer-Verlag, pp. 388–397, 1999.
13. Chae Hoon Lim, *Crypton: a new 128-bit Block Cipher - Specifications and Analysis*, submitted to the Advanced Encryption Standard (AES) contest, 1998.
14. James L. Massey, Gurgun H. Khachatrian, Melsik K. Kuregian, *Nomination of Safer++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption(NESSIE)*, submitted to NESSIE, 2000.
15. Mitsuru Matsui, *Linear cryptanalysis method for DES cipher*, Advances in Cryptology, proceedings of Eurocrypt'93, Lecture Notes in Computer Science 765, T. Helleseeth, Ed., Springer-Verlag, pp. 386–397, 1994.
16. Nippon Telegraph and Telephone Corporation, *AES Proposal: E2*, submitted to the Advanced Encryption Standard (AES) contest, 1998.
17. National Institute of Standards and Technology, *FIPS-197:Advanced Encryption Standard*, Federal Information Processing Standard, FIPS-197, 2001.
18. Ronald L. Rivest, Adi Shamir, Leonard Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, 21(2):120–126, 1978.
19. Serge Vaudenay, *Alert on Non-Linearity: Linearities in RIJNDAEL, KASUMI,...*, presented in the rump session of Crypto'01.

## A The Affine Transformation of Rijndael and Rijndael<sup>2</sup>

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad A^2 = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

# On the Security of Rijndael-Like Structures against Differential and Linear Cryptanalysis

Sangwoo Park<sup>1</sup>, Soo Hak Sung<sup>2</sup>, Seongtaek Chee<sup>1</sup>,  
E-Joong Yoon<sup>1</sup>, and Jongin Lim<sup>3</sup>

<sup>1</sup> National Security Research Institute, Korea,  
{psw,chee,yej}@etri.re.kr

<sup>2</sup> Department of Applied Mathematics, Pai Chai University, Korea,  
sungsh@woonam.paichai.ac.kr

<sup>3</sup> Center for Information Security Technologies(CIST), Korea University, Korea,  
jilim@cist.korea.ac.kr

**Abstract.** Rijndael-like structure is a special case of SPN structure. The linear transformation of Rijndael-like structures consists of linear transformations of two types, the one is byte permutation  $\pi$  and the other is linear transformation  $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ , where each of  $\theta_i$  separately operates on each of the four columns of a state. Furthermore,  $\pi$  and  $\theta$  have some interesting properties. In this paper, we present a new method for upper bounding the maximum differential probability and the maximum linear hull probability for Rijndael-like structures. By applying our method to Rijndael, we obtain that the maximum differential probability and the maximum linear hull probability for 4 rounds of Rijndael are bounded by  $1.06 \times 2^{-96}$ .

## 1 Introduction

SPN(Substitution and Permutation Network) structure is one of the most commonly used structure in block ciphers. SPN structure is based on Shannon's principles of confusion and diffusion [4] and these principles are implemented through the use of substitution and linear transformation, respectively.

Rijndael [7], Crypton [12,13] and Square [6] are the block ciphers composed of SPN structures. They have a common point for the type of their linear transformations. Each of their linear transformations consists of linear transformations of two types, the one is byte permutation  $\pi$  and the other is linear transformation  $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ , where each of  $\theta_i$  separately operates on each of the four columns of a state. Furthermore, each of bytes of each column of  $y = \pi(x)$  comes from each different column of  $x$ , and we can determine the branch number of each of  $\theta_i$ . In this paper, we call such a SPN structure Rijndael-like structure.

The security of SPN structures against differential cryptanalysis [2,3] and linear cryptanalysis [14] depends on the maximum differential probability and the maximum linear hull probability. In [11], Keliher *et al.* proposed a method for finding the upper bound on the maximum average linear hull probability for SPN structures. Application of their method to Rijndael yields an upper bound

of  $2^{-75}$  when 7 or more rounds are approximated. In [10], it was proposed that the improved upper bound on the maximum average linear hull probability for Rijndael when 9 or more rounds are approximated is  $2^{-92}$ , corresponding to a lower bound on the data complexity of  $2^{97}$ . This is based on completion of 43% of the computation. It is estimated that the running time to completion is 200,000 hours on a single Sun Ultra 5.

In this paper, we present a new method for upper bounding the maximum differential probability and the maximum linear hull probability for Rijndael-like structures. We prove that the maximum differential probability for 4 rounds of Rijndael-like structures is bounded by  $4p^{19} + 6p^{18} + 4p^{17} + p^{16}$ , when the maximum differential probability for S-boxes is  $p(\leq 2^{-3})$ . Also, we prove that the maximum linear hull probability for 4 rounds of Rijndael-like structures is bounded by  $4q^{19} + 6q^{18} + 4q^{17} + q^{16}$ , when the maximum linear hull probability for S-boxes is  $q(\leq 2^{-3})$ . By applying our method to Rijndael, we obtain that the maximum differential probability and the maximum linear hull probability for 4 rounds of Rijndael are bounded by  $1.06 \times 2^{-96}$ .

## 2 SPN Structures

One round of SPN structures generally consists of three layers of key addition, substitution, and linear transformation. On the key addition layer, round subkeys and round input values are exclusive-ored. Substitution layer is made up of  $n$  small nonlinear substitutions referred to as S-boxes, and linear transformation layer is a linear transformation in order to diffuse the cryptographic characteristics of substitution layer. A typical example of one round of SPN structures is given in Figure 1.

On  $r$  rounds of SPN structures, the linear transformation of the last round, generally, is omitted, because it has no cryptographic significance. Therefore, 2 rounds of SPN structures is given in Figure 2.

S-boxes and linear transformations should be invertible in order to decipher. Therefore we assume that all S-boxes are bijections from  $Z_2^m$  to itself. Moreover, throughout this paper, we assume that round subkeys are independent and uniformly distributed.

Let  $S$  be an S-box with  $m$  input and output bits. Differential and linear probability of  $S$  are defined as the following definition:

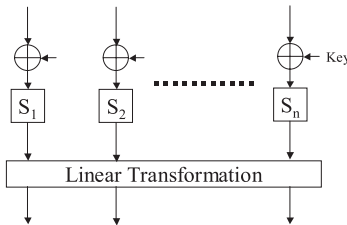
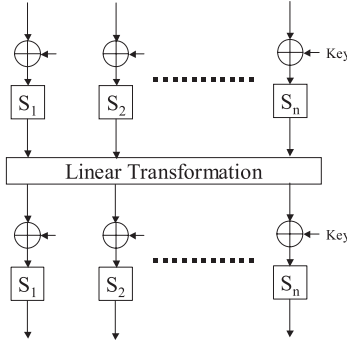


Fig. 1. One round of SPN structures



**Fig. 2.** 2 rounds of SPN structures

**Definition 1.** For any given  $a, b, \Gamma_a, \Gamma_b \in \mathbb{Z}_2^m$ , define differential probability  $DP^S(a, b)$  and linear probability  $LP^S(\Gamma_a, \Gamma_b)$  of  $S$  by

$$DP^S(a, b) = \frac{\#\{x \in \mathbb{Z}_2^m | S(x) \oplus S(x \oplus a) = b\}}{2^m}$$

and

$$LP^S(\Gamma_a, \Gamma_b) = \left( \frac{\#\{x \in \mathbb{Z}_2^m | \Gamma_a \cdot x = \Gamma_b \cdot S(x)\}}{2^{m-1}} - 1 \right)^2,$$

respectively, where  $x \cdot y$  denotes the parity (0 or 1) of bitwise product of  $x$  and  $y$ .

$a$  and  $b$  are called as input and output differences, respectively. Also,  $\Gamma_a$  and  $\Gamma_b$  are called as input and output mask values, respectively.

The strength of an S-box  $S$  against differential cryptanalysis is decided by maximum differential probability  $\max_{a \neq 0, b} DP^S(a, b)$ . The strength of an S-box  $S$  against linear cryptanalysis is decided by maximum linear probability  $\max_{\Gamma_a, \Gamma_b \neq 0} LP^S(\Gamma_a, \Gamma_b)$ .

**Definition 2.** The maximum differential probability  $p$  and the maximum linear probability  $q$  of  $S$  are defined by

$$p = \max_{a \neq 0, b} DP^S(a, b)$$

and

$$q = \max_{\Gamma_a, \Gamma_b \neq 0} LP^S(\Gamma_a, \Gamma_b),$$

respectively.

The maximum differential probability  $p$  and the maximum linear probability  $q$  for a strong S-box  $S$  should be small enough for any input difference  $a \neq 0$  and any output mask value  $\Gamma_b \neq 0$ .

**Definition 3.** Differentially active S-box is defined as an S-box given a non-zero input difference and linearly active S-box is defined as an S-box given a nonzero output mask value.

Since all S-boxes in substitution layer are bijective, if an S-box is differentially/linearly active, then it has a non-zero output difference/input mask value.

For SPN structures, between the differential probability and the number of differentially active S-boxes, there is a relationship which is close. When the number of differentially active S-boxes is many, the differential probability comes to be small, and when the number of differentially active S-boxes is small, the differential probability comes to be big. Therefore, the concept of the branch number was proposed [6]. We call it the branch number from the viewpoint of differential cryptanalysis, the minimum number of differentially active S-boxes of 2 rounds of SPN structures. Also, we call it the branch number from the viewpoint of linear cryptanalysis, the minimum number of linearly active S-boxes of 2 rounds of SPN structures.

The linear transformation  $L : (Z_2^m)^n \rightarrow (Z_2^m)^n$  can be represented by  $n \times n$  matrix  $M = (m_{ij})$  and  $L(x) = Mx$ , where  $x \in (Z_2^m)^n$  and the addition is bitwise exclusive-ored. For the block cipher E2 [15] and Camellia [11],  $m_{ij} \in Z_2$  and the multiplication is trivial. For the block cipher Crypton [12][13],  $m_{ij} \in Z_2^m$  and the multiplication is the bitwise logical-and operation. For the block cipher Rijndael [7],  $m_{ij} \in GF(2^m)$  and the multiplication is defined as the multiplication over  $GF(2^m)$ .

It is easy to show that  $L(x) \oplus L(x^*) = L(x \oplus x^*)$  and  $DP^L(a, L(a)) = 1$  [5].

**Definition 4.** Let  $L$  be the linear transformation over  $(Z_2^m)^n$ . The branch number of  $L$  from the view point of differential cryptanalysis,  $\beta_d$ , is defined by

$$\beta_d = \min_{x \neq 0} \{wt(x) + wt(L(x))\},$$

where,  $wt(x) = wt(x_1, x_2, \dots, x_n) = \#\{1 \leq i \leq n | x_i \neq 0\}$ .

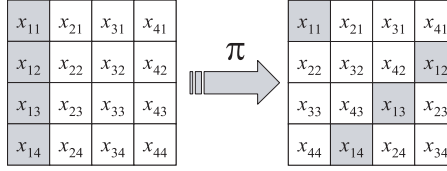
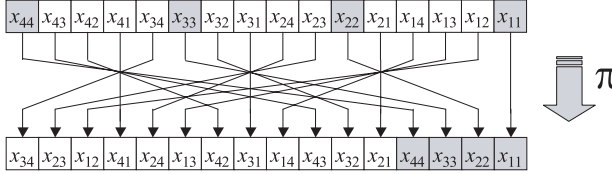
Throughout this paper, we define  $wt(x) = wt(x_1, x_2, \dots, x_n) = \#\{1 \leq i \leq n | x_i \neq 0\}$  when  $x = (x_1, x_2, \dots, x_n)$ . If  $x \in Z_2^m$ , then  $wt(x)$  is the Hamming weight of  $x$ .

It is proved that, if  $m_{ij} \in Z_2$ , then  $LP^L(M^t \Gamma_b, \Gamma_b) = 1$ . Therefore, we know that  $LP^L(\Gamma_a, (M^{-1})^t \Gamma_a) = 1$ . Also, if  $m_{ij} \in GF(2^m)$ , then it is proved that  $LP^L(\Gamma_a, C\Gamma_a) = 1$ , for some  $n \times n$  matrix  $C$  over  $GF(2^m)$  [9]. Therefore, we can define the branch number  $\beta_l$  from the view point of linear cryptanalysis as follows:

$$\beta_l = \begin{cases} \min_{\Gamma_a \neq 0} \{wt(\Gamma_a) + wt((M^{-1})^t \Gamma_a)\}, & \text{if } m_{ij} \in Z_2, 1 \leq i, j \leq n, \\ \min_{\Gamma_a \neq 0} \{wt(\Gamma_a) + wt(C\Gamma_a)\}, & \text{if } m_{ij} \in GF(2^m), 1 \leq i, j \leq n. \end{cases}$$

### 3 Rijndael-Like Structures

Rijndael is the block cipher composed of SPN structures and its linear transformation consists of ShiftRows transformation and MixColumns transformation. We analyze some interesting properties of ShiftRows transformation and MixColumns transformation of Rijndael.

**Fig. 3.** ShiftRows transformation of Rijndael**Fig. 4.** Another representation of ShiftRows transformation of Rijndael

Let  $\pi : (Z_2^8)^{16} \rightarrow (Z_2^8)^{16}$  be the ShiftRows transformation of Rijndael. Let  $x = (x_1, x_2, x_3, x_4) = (x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, \dots, x_{34}, x_{41}, x_{42}, x_{43}, x_{44})$  be the input of  $\pi$ . Figure 3 and 4 illustrate the ShiftRows transformation  $\pi$  of Rijndael.

Let  $y = (y_1, y_2, y_3, y_4) = (y_{11}, y_{12}, y_{13}, y_{14}, y_{21}, \dots, y_{34}, y_{41}, y_{42}, y_{43}, y_{44})$  be the output of  $\pi$ . It is easy to know that, for any  $i (i = 1, 2, 3, 4)$ , each of bytes of  $y_i$  comes from each different  $x_i$ . For example, for  $y_1 = (y_{11}, y_{12}, y_{13}, y_{14}) = (x_{11}, x_{22}, x_{33}, x_{44})$ ,  $x_{11}$  is a byte coming from  $x_1$ . Furthermore,  $x_{22}$ ,  $x_{33}$  and  $x_{44}$  are elements of  $x_2$ ,  $x_3$  and  $x_4$ , respectively.

The MixColumns transformation of Rijndael operates on the state column by column, treating each column as a four-term polynomial. Let  $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$  be the MixColumns transformation of Rijndael. Let  $y = (y_1, y_2, y_3, y_4) = (y_{11}, y_{12}, y_{13}, y_{14}, y_{21}, \dots, y_{34}, y_{41}, y_{42}, y_{43}, y_{44})$  be the input of  $\theta$  and  $z = (z_1, z_2, z_3, z_4) = (z_{11}, z_{12}, z_{13}, z_{14}, z_{21}, \dots, z_{34}, z_{41}, z_{42}, z_{43}, z_{44})$  be the output of  $\theta$ , respectively. Each of  $\theta_i$  can be written as a matrix multiplication as follows:

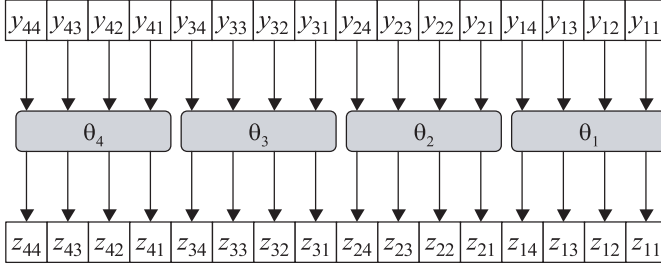
$$\begin{pmatrix} y_{i1} \\ y_{i2} \\ y_{i3} \\ y_{i4} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} z_{i1} \\ z_{i2} \\ z_{i3} \\ z_{i4} \end{pmatrix}.$$

In the matrix multiplication, the addition is bitwise exclusive-ored and the multiplication is defined as the multiplication over  $GF(2^8)$ . Figure 5 illustrates the MixColumns transformation  $\theta$  of Rijndael. We can consider each of  $\theta_i$  as a linear transformation and we know that the branch number of each of  $\theta_i$  is 5.

**Definition 5.** *Rijndael-like structures are the block ciphers composed of SPN structures satisfying the followings:*

- (i) *Their linear transformation has the form  $(\theta_1, \theta_2, \theta_3, \theta_4) \circ \pi$ .*





**Fig. 5.** The MixColumns transformation of Rijndael

(ii) (The condition of  $\pi$ ) Each of bytes of  $y_i$  comes from each different  $x_i$ , where  $x = (x_1, x_2, x_3, x_4)$  is input of  $\pi$  and  $y = (y_1, y_2, y_3, y_4)$  is output of  $\pi$ , respectively.

(iii) (The condition of  $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ ) When we consider each of  $\theta_i$  as a linear transformation, the followings hold:

$$\beta_d^{\theta_1} = \beta_d^{\theta_2} = \beta_d^{\theta_3} = \beta_d^{\theta_4} \text{ and } \beta_l^{\theta_1} = \beta_l^{\theta_2} = \beta_l^{\theta_3} = \beta_l^{\theta_4}.$$

Rijndael, Square and Crypton are examples of Rijndael-like structures.

**Definition 6.** For  $x = (x_1, \dots, x_n)$ , the pattern of  $x$ ,  $\gamma_x$ , is defined by  $\gamma_x = (\gamma_1, \dots, \gamma_n) \in \mathbb{Z}_2^n$ , where, if  $x_i = 0$ , then  $\gamma_i = 0$ , and if  $x_i \neq 0$ , then  $\gamma_i = 1$ .

If  $x = (x_1, x_2, x_3, x_4)$ , where  $x_1 \neq 0$ ,  $x_2 \neq 0$  and  $x_3 = x_4 = 0$ , then  $\gamma_x = (1, 1, 0, 0)$ .

**Definition 7.** Let  $x = (x_1, x_2, x_3, x_4)$  be the input of  $\pi$  and  $y = (y_1, y_2, y_3, y_4)$  be the output of  $\pi$ , respectively. For arbitrary  $\gamma \in \mathbb{Z}_2^4$  and  $u = (u_1, u_2, u_3, u_4) \in \mathbb{Z}^4$ , We define  $N[\gamma, u]$  as following:

$$N[\gamma, u] = \#\{y = \pi(x) | \gamma_x = \gamma, wt(y_i) = u_i, 1 \leq i \leq 4\}.$$

$N[\gamma, u]$  means the number of  $y = \pi(x)$  such that  $wt(y_i) = u_i (1 \leq i \leq 4)$ , when the pattern of input of  $\pi$  is  $\gamma$ .  $N[\gamma, u]$  is well-defined and, for any linear transformation which satisfies the condition of  $\pi$ , the values of  $N[\gamma, u]$  are all the same for some fixed  $\gamma$  and  $u = (u_1, u_2, u_3, u_4)$ . The followings are the main properties of  $N[\gamma, u]$ :

- For some  $i$ , if  $u_i > wt(\gamma)$ , then  $N[\gamma, u] = 0$ , because  $wt(y_i) \leq wt(\gamma_x)$ .
- If  $u_1 + u_2 + u_3 + u_4 < wt(\gamma)$ , then  $N[\gamma, u] = 0$ , because  $\sum_{i=1}^4 wt(y_i) = \sum_{i=1}^4 wt(x_i) \geq wt(\gamma_x)$ .
- If  $\max\{u_1, \dots, u_4\} = wt(\gamma)$ , then  $N[\gamma, u] = \binom{wt(\gamma)}{u_1} \cdots \binom{wt(\gamma)}{u_4}$ .
- For any permutation  $\phi$  and  $\rho$  over  $\{1, 2, 3, 4\}$ ,

$$\begin{aligned} & N[(\gamma_1, \gamma_2, \gamma_3, \gamma_4), (u_1, u_2, u_3, u_4)] \\ &= N[(\gamma_{\phi(1)}, \gamma_{\phi(2)}, \gamma_{\phi(3)}, \gamma_{\phi(4)}), (u_{\rho(1)}, u_{\rho(2)}, u_{\rho(3)}, u_{\rho(4)})] \end{aligned}$$

*Example 1.* For some  $\gamma$  and  $u$ , it is easy to determine the value of  $N[\gamma, u]$ . The followings are the examples:

- $N[(1, 1, 1, 0), (4, 1, 0, 0)] = 0$ .
- $N[(1, 1, 1, 0), (1, 1, 0, 0)] = 0$ .
- $N[(1, 1, 1, 0), (3, 2, 2, 0)] = 9$ .
- $N[(1, 1, 1, 0), (2, 1, 0, 0)] = 3$ .

## 4 The Upper Bound on the Differential and the Linear Hull Probabilities for Rijndael-Like Structures

To compute the upper bound on the maximum differential probability for  $r(r \geq 2)$  rounds of Rijndael-like structures, we assume the following:

$$\beta_d^{\theta_1} = \beta_d^{\theta_2} = \beta_d^{\theta_3} = \beta_d^{\theta_4} = 5 \text{ and } \beta_l^{\theta_1} = \beta_l^{\theta_2} = \beta_l^{\theta_3} = \beta_l^{\theta_4} = 5.$$

and we need the following notations:

- $a = (a_1, \dots, a_4) = (a_{11}, a_{12}, a_{13}, a_{14}, \dots, a_{41}, a_{42}, a_{43}, a_{44})$ : input difference.
- $b = (b_1, \dots, b_4) = (b_{11}, b_{12}, b_{13}, b_{14}, \dots, b_{41}, b_{42}, b_{43}, b_{44})$ : output difference.
- $DP_r(a, b)$ : differential probability of  $r$  rounds whose input difference is  $a$  and output difference is  $b$ .
- $x^{(i)} = (x_1^{(i)}, \dots, x_4^{(i)}) = (x_{11}^{(i)}, x_{12}^{(i)}, x_{13}^{(i)}, x_{14}^{(i)}, \dots, x_{41}^{(i)}, x_{42}^{(i)}, x_{43}^{(i)}, x_{44}^{(i)})$ : the input of  $\pi$  at  $i$ -th round.
- $y^{(i)} = (y_1^{(i)}, \dots, y_4^{(i)}) = (y_{11}^{(i)}, y_{12}^{(i)}, y_{13}^{(i)}, y_{14}^{(i)}, \dots, y_{41}^{(i)}, y_{42}^{(i)}, y_{43}^{(i)}, y_{44}^{(i)})$ : the output of  $\pi$  at  $i$ -th round, i.e. the input of  $\theta$  at  $i$ -th round.
- $z^{(i)} = (z_1^{(i)}, \dots, z_4^{(i)}) = (z_{11}^{(i)}, z_{12}^{(i)}, z_{13}^{(i)}, z_{14}^{(i)}, \dots, z_{41}^{(i)}, z_{42}^{(i)}, z_{43}^{(i)}, z_{44}^{(i)})$ : the output of  $\theta$  at  $i$ -th round.

When the branch number is  $n$  or  $n + 1$ , it is known that the upper bounds of the maximum differential probability and the linear hull probability for 2 rounds of SPN structures are as follows:

**Lemma 1** ([8,9]).

- If  $\beta_d = n + 1$  or  $n$ , then  $DP_2(a, b) \leq p^{\beta_d - 1}$ .
- If  $\beta_l = n + 1$  or  $n$ , then  $LP_2(\Gamma_a, \Gamma_b) \leq q^{\beta_l - 1}$ .

The upper bound on the maximum differential probability for 2 rounds of Rijndael-like structures is obtained by Lemma 1.

**Theorem 1.**

$$DP_2(a, b) \leq \begin{cases} p^{wt(\gamma_{\pi(a)})(\beta_d - 1)}, & \text{if } \gamma_{\pi(a)} = \gamma_b, \\ 0, & \text{otherwise.} \end{cases}$$

*Proof.* Let  $\pi(a) = (a_1^*, a_2^*, a_3^*, a_4^*)$ . Then  $DP_2(a, b) = \prod_{i=1}^4 DP_2^{\theta_i}(a_i^*, b_i)$ , where,  $DP_2^{\theta_i}$  is the differential probability of 2 rounds of SPN structure whose linear transformation is  $\theta_i$ . By Lemma 1, we know that the upper bound on  $DP_2^{\theta_i}(a_i^*, b_i)$  is the followings:

$$DP_2^{\theta_i}(a_i^*, b_i) \leq \begin{cases} p^{\beta_d-1}, & \text{if } a_i^* \neq 0, b_i \neq 0, \\ 1, & \text{if } a_i^* = 0, b_i = 0, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the proof is completed.

By Theorem 1 the upper bound on the maximum differential probability for 2 rounds of Rijndael-like structures is  $p^{\beta_d-1}$ . By applying Theorem 1 to Rijndael, we obtain that the maximum differential probability for 2 rounds of Rijndael is bounded by  $2^{-24}$ , because  $\beta_d = 5$ ,  $p = 2^{-6}$ .

Now, we compute the upper bound on the maximum differential probability for 3 rounds of Rijndael-like structures. To do this, we prove the following:

**Lemma 2.** Let  $L : (Z_2^m)^n \rightarrow (Z_2^m)^n$  be the linear transformation whose branch number is  $\beta_d$ . For  $\gamma \in Z_2^n$  and  $b = (b_1, \dots, b_n) \in (Z_2^m)^n$  with  $wt(\gamma) + wt(b) \geq \beta_d$ , we define the set  $A$  as following:

$$A = \{y = (y_1, \dots, y_n) \in (Z_2^m)^n \mid \gamma_y = \gamma_b, y = L(x) \text{ for some } x \text{ such that } \gamma_x = \gamma\}.$$

Then, the following holds:

$$\sum_{y \in A} DP_1(y, b) = \sum_{y \in A} DP(y_1, b_1) \cdots DP(y_n, b_n) \leq p^{\max\{0, \beta_d - wt(\gamma) - 1\}}$$

*Proof.* Since  $\sum_{y \in A} DP_1(y, b) \leq \sum_{y \in (Z_2^m)^n} DP_1(y, b) = 1$ , it is sufficient to consider the case  $\beta_d - wt(\gamma) - 1 > 0$ . Without loss of generality, we assume that  $wt(b) = k$  and  $b_1 \neq 0, \dots, b_k \neq 0, b_{k+1} = \dots = b_n = 0$ . Then

$$\sum_{y \in A} DP_1(y, b) = \sum_{y \in A} DP(y_1, b_1) \cdots DP(y_k, b_k). \quad (1)$$

We proceed the proof with two cases:  $wt(\gamma) + wt(b) = \beta_d$  and  $wt(\gamma) + wt(b) > \beta_d$ .

(Case 1:  $wt(\gamma) + wt(b) = \beta_d$ ). For any  $i (1 \leq i \leq k)$ , let  $y_{i,1}, y_{i,2}, \dots, y_{i,\delta}$  be all possible values of  $y_i$  in Equation (1). Then, for each  $i (1 \leq i \leq k)$ ,  $y_{i,1}, y_{i,2}, \dots, y_{i,\delta}$  are distinct, because  $L$  is linear and  $wt(\gamma) + wt(b) = \beta_d$ . If, for some  $i (1 \leq i \leq k)$ ,  $y_{i,1}, y_{i,2}, \dots, y_{i,\delta}$  are not distinct, then there exist a pair  $(y_{i,l}, y_{i,l'})$  such that  $y_{i,l} = y_{i,l'}$ , where  $y_{i,l}$  is  $i$ -th component of  $y = L(x)$  and  $y_{i,l'}$  is  $i$ -th component of  $y' = L(x')$ , respectively. Since  $L(x) \oplus L(x') = L(x \oplus x')$ ,  $i$ -th component of  $L(x \oplus x')$  is equal to zero. This is a contradiction of the definition of branch number. Therefore, we can establish the following:

$$\sum_{y \in A} DP_1(y, b) \leq p^{k-1} \sum_{y \in A} DP(y_1, b_1) \leq p^{k-1} = p^{\beta_d - wt(\gamma) - 1}.$$

(Case 2:  $wt(\gamma) + wt(b) > \beta_d$ ). In this case,  $y_{i,1}, y_{i,2}, \dots, y_{i,\delta}$  are not necessarily distinct. We fix  $t = k + wt(\gamma) - \beta_d$  components of nonzero components of  $y$ , i.e.,  $y_1, y_2, \dots, y_t$ . Then, all possible values of each of another components ( $y_{t+1}, \dots, y_k$ ) are distinct. Therefore, we can establish the following:

$$\begin{aligned}
& \sum_{y \in A} DP_1(y, b) \\
& \leq \sum_{j_1=1}^{2^m-1} DP(j_1, b_1) \cdots \sum_{j_t=1}^{2^m-1} DP(j_t, b_t) \sum_{y \in A, y_i=j_i, 1 \leq i \leq t} DP(y_{t+1}, b_{t+1}) \cdots DP(y_k, b_k) \\
& \leq p^{k-t-1} \sum_{j_1=1}^{2^m-1} DP(j_1, b_1) \cdots \sum_{j_t=1}^{2^m-1} DP(j_t, b_t) \sum_{y \in A, y_i=j_i, 1 \leq i \leq t} DP(y_{t+1}, b_{t+1}) \\
& \leq p^{k-t-1} \sum_{j_1=1}^{2^m-1} DP(j_1, b_1) \cdots \sum_{j_t=1}^{2^m-1} DP(j_t, b_t) \\
& = p^{k-t-1} = p^{\beta_d - wt(\gamma) - 1}.
\end{aligned}$$

**Theorem 2.** Let  $wt(\gamma_{\pi(a)}) = l$  and  $wt(b) = k$ . Let  $b_{t_1}, \dots, b_{t_k}$  be the nonzero components of  $b = (b_1, b_2, b_3, b_4)$ . Then

$$\begin{aligned}
& DP_3(a, b) \\
& \leq p^{l(\beta_d-1)} \sum_{j_1=\beta_d-wt(b_{t_1})}^l \cdots \sum_{j_k=\beta_d-wt(b_{t_k})}^l N[\gamma_{\pi(a)}, (u_1, u_2, u_3, u_4)] \cdot p^{\sum_{i=1}^k \max\{0, \beta_d - j_i - 1\}},
\end{aligned}$$

where, each of  $u_i (1 \leq i \leq 4)$  is the following:

$$u_i = \begin{cases} j_s, & \text{if } i = t_s \text{ for some } t_s \\ 0, & \text{otherwise.} \end{cases}$$

*Proof.* Without loss of generality, we assume that  $t_1 = 1, \dots, t_k = k$ . By Theorem [1](#)

$$\begin{aligned}
DP_3(a, b) &= \sum_{x^{(2)}} DP_2(a, x^{(2)}) DP_1(z^{(2)}, b) \\
&= \sum_{\gamma_{x^{(2)}} = \gamma_{\pi(a)}} DP_2(a, x^{(2)}) DP_1(z^{(2)}, b) \\
&\leq \max_{\gamma_{x^{(2)}} = \gamma_{\pi(a)}} DP_2(a, x^{(2)}) \sum_{z^{(2)}} DP_1(z^{(2)}, b) \\
&\leq p^{l(\beta_d-1)} \sum_{z^{(2)}} DP_1(z^{(2)}, b),
\end{aligned}$$

where,  $z^{(2)} = L(x^{(2)}) = (\theta_1(y_1^{(2)}), \theta_2(y_2^{(2)}), \theta_3(y_3^{(2)}), \theta_4(y_4^{(2)}))$ . Furthermore, the following three conditions hold:

- (i)  $\gamma_{z_1^{(2)}} = \gamma_{b_1}, \dots, \gamma_{z_k^{(2)}} = \gamma_{b_k}, \gamma_{z_{k+1}^{(2)}} = \dots = \gamma_{z_4^{(2)}} = 0,$
- (ii)  $y_1^{(2)} \neq 0, \dots, y_k^{(2)} \neq 0, y_{k+1}^{(2)} = \dots = y_4^{(2)} = 0,$
- (iii)  $\gamma_{x^{(2)}} = \gamma_{\pi(a)}.$

For each  $i (1 \leq i \leq k)$ , since  $y_i^{(2)}$  and  $z_i^{(2)}$  are the nonzero input and nonzero output of  $\theta_i$ , respectively, we know that  $wt(y_i^{(2)}) + wt(z_i^{(2)}) \geq \beta_d$ . Furthermore, since  $wt(b_i) = wt(z_i^{(2)})$ , we know that  $wt(y_i^{(2)}) + wt(b_i) \geq \beta_d$ . Therefore, since  $wt(y_i^{(2)}) \leq wt(\gamma_{x^{(2)}})$ , we can establish the following equation:

$$\beta_d - wt(b_i) \leq wt(y_i^{(2)}) \leq wt(\gamma_{\pi(a)}), 1 \leq i \leq k.$$

Now, we consider  $j_i (1 \leq i \leq k)$  such that  $\beta_d - wt(b_i) \leq j_i \leq wt(\gamma_{\pi(a)})$ . For  $(\gamma_1, \dots, \gamma_4)$  which satisfies that if  $1 \leq i \leq k$ , then  $wt(\gamma_i) = j_i$  and if  $k+1 \leq i \leq 4$ , then  $wt(\gamma_i) = 0$ , we define the set  $A_{(\gamma_1, \dots, \gamma_4)}$  as following:

$$A_{(\gamma_1, \dots, \gamma_4)} = \{z^{(2)} = (z_1^{(2)}, \dots, z_4^{(2)}) | \gamma_{y_i^{(2)}} = \gamma_i, 1 \leq i \leq 4\}$$

where,  $z^{(2)}$  satisfies the three conditions (i), (ii) and (iii).

The set  $A_{(\gamma_1, \dots, \gamma_4)}$  can be empty set, but, the number of non-empty set  $A_{(\gamma_1, \dots, \gamma_4)}$  is  $N[\gamma_{\pi(a)}, (j_1, \dots, j_k, 0, \dots, 0)]$ . If  $A_{(\gamma_1, \dots, \gamma_4)}$  is not empty set, by Lemma 2,

$$\begin{aligned} \sum_{z^{(2)} \in A_{(\gamma_1, \dots, \gamma_4)}} DP_1(z^{(2)}, b) &= \sum_{z_1^{(2)}} DP_1(z_1^{(2)}, b_1) \cdots \sum_{z_k^{(2)}} DP_1(z_k^{(2)}, b_k) \\ &\leq \prod_{i=1}^k p^{\max\{0, \beta_d - j_i - 1\}}. \end{aligned}$$

Therefore,

$$\begin{aligned} &\sum_{z^{(2)}} DP_1(z^{(2)}, b) \\ &\leq \sum_{j_1 = \beta_d - wt(b_{t_1})}^l \cdots \sum_{j_k = \beta_d - wt(b_{t_k})}^l N[\gamma_{\pi(a)}, (u_1, u_2, u_3, u_4)] \cdot p^{\sum_{i=1}^k \max\{0, \beta_d - j_i - 1\}}. \end{aligned}$$

Therefore, the proof is completed.

To derive the upper bound on the maximum differential probability for 4 rounds of Rijndael-like structures, we prove the following three lemmas:

**Lemma 3.** *If  $wt(\gamma_{\pi(a)}) = 2, wt(b) = 3$ , then  $DP_4(a, b) \leq 4p^{19} + 6p^{18} + 4p^{17} + p^{16}$ .*

*Proof.* We assume that  $\gamma_b = (1, 1, 1, 0)$ . Then we can represent  $DP_4(a, b)$  as following:

$$\begin{aligned} DP_4(a, b) &= \sum_{x^{(3)}} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &= \sum_{i=1}^4 \sum_{wt(x^{(3)})=i} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &:= I + II + III + IV. \end{aligned}$$

We know that  $wt(y_i^{(2)}) \leq wt(x^{(2)}) = wt(\gamma_{\pi(a)}) = 2$  and  $wt(z_i^{(2)}) = wt(x_i^{(3)}) \leq wt(b) = 3$ . Since  $\beta_d^{\theta_i} = 5$ ,  $wt(x_i^{(3)}) = 3$ , where  $x_i^{(3)}$  is nonzero component of  $x^{(3)}$ . Now, we compute the value of  $I$ . We can represent  $I$  as following:

$$\begin{aligned} I &= \sum_{\gamma_{x(3)}=(1,0,0,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) + \sum_{\gamma_{x(3)}=(0,1,0,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &+ \sum_{\gamma_{x(3)}=(0,0,1,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) + \sum_{\gamma_{x(3)}=(0,0,0,1)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &:= I_1 + I_2 + I_3 + I_4 \end{aligned}$$

At first, we compute the value of  $I_1$ . Since  $wt(x_1^{(3)}) = 3$ , by Theorem 2,

$$\max_{\gamma_{x(3)}=(1,0,0,0)} DP_3(a, x^{(3)}) \leq p^8 \sum_{j=2}^2 N[\gamma_{\pi(a)}, (j, 0, 0, 0)] p^{4-j} = p^{10}.$$

Since  $wt(x_1^{(3)}) = 3$  and  $\gamma_b = (1, 1, 1, 0)$ , the number of patterns,  $(y_1^{(3)}, y_2^{(3)}, y_3^{(3)}, 0)$  is equal to  $N[(1, 1, 1, 0), (3, 0, 0, 0)] = 1$ . For the pattern  $(\gamma_1, \gamma_2, \gamma_3, 0)$ , by Lemma 2,

$$\begin{aligned} &\sum_{\gamma_{x(3)}=(1,0,0,0)} DP_1(z^{(3)}, b) \\ &= \sum_{\gamma_{y_1^{(3)}}=\gamma_1} DP_1(z_1^{(3)}, b_1) \sum_{\gamma_{y_2^{(3)}}=\gamma_2} DP_2(z_2^{(3)}, b_2) \sum_{\gamma_{y_3^{(3)}}=\gamma_3} DP_3(z_3^{(3)}, b_3) \\ &\leq p^{12-(wt(\gamma_1)+wt(\gamma_2)+wt(\gamma_3))} \leq p^9. \end{aligned}$$

Therefore,

$$I_1 \leq \max_{\gamma_{x(3)}=(1,0,0,0)} DP_3(a, x^{(3)}) \sum_{\gamma_{x(3)}=(1,0,0,0)} DP_1(z^{(3)}, b) \leq p^{19}.$$

By applying the same method, it can be determined that the upper bounds of  $I_2$ ,  $I_3$  and  $I_4$  are the same with that of  $I_1$ . Therefore, we arrive at  $I \leq 4p^{19}$ . Furthermore, using the same method, we have that  $II \leq 6p^{18}$  and  $III \leq 4p^{17}$ . At last, the upper bound on  $IV$  can be computed by Theorem 3 as follows:

$$\begin{aligned} IV &\leq \max_{wt(x^{(3)})=4} DP_3(a, x^{(3)}) \\ &= \max_{wt(x^{(3)})=4} \sum_{x^{(1)}} DP_1(a, x^{(1)}) DP_2(z^{(1)}, x^{(3)}) \\ &\leq \max_{wt(x^{(3)})=4} \max_{z^{(1)}} DP_2(z^{(1)}, x^{(3)}) \leq p^{16}. \end{aligned}$$

Therefore,

$$DP_4(a, b) = I + II + III + IV \leq 4p^{19} + 6p^{18} + 4p^{17} + p^{16}.$$

**Lemma 4.** *If  $wt(\gamma_{\pi(a)}) = 3$ ,  $wt(b) = 2$ , then  $DP_4(a, b) \leq 4p^{19} + 6p^{18} + 4p^{17} + p^{16}$ .*

*Proof.* The proof is similar to that of Lemma 3 and is omitted.

**Lemma 5.** *If  $wt(\gamma_{\pi(a)}) = 3$ ,  $wt(b) = 3$ , then  $DP_4(a, b) \leq 184p^{22} + 912p^{21} + 438p^{20} + 72p^{19} + 4p^{18} + p^{16}$ .*

*Proof.* We assume that  $\gamma_b = (1, 1, 1, 0)$ . Then we can represent  $DP_4(a, b)$  as following:

$$\begin{aligned} DP_4(a, b) &= \sum_{x^{(3)}} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &= \sum_{i=1}^4 \sum_{wt(x^{(3)})=i} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &:= I + II + III + IV. \end{aligned}$$

We know that  $wt(y_i^{(2)}) \leq wt(x^{(2)}) = wt(\gamma_{\pi(a)}) = 3$  and  $wt(z_i^{(2)}) = wt(x_i^{(3)}) \leq wt(b) = 3$ . Since  $\beta_d^{\theta_i} = 5$ ,  $wt(x_i^{(3)}) = 2$  or  $3$ , where  $x_i^{(3)}$  is nonzero component of  $x^{(3)}$ . Now, we compute the value of  $I$ . We can represent  $I$  as follows:

$$\begin{aligned} I &= \sum_{\gamma_{x(3)}=(1,0,0,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) + \sum_{\gamma_{x(3)}=(0,1,0,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &\quad + \sum_{\gamma_{x(3)}=(0,0,1,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) + \sum_{\gamma_{x(3)}=(0,0,0,1)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &:= I_1 + I_2 + I_3 + I_4. \end{aligned}$$

At first, we compute the value of  $I_1$ . Since  $\sum_{i=1}^4 wt(x_i^{(3)}) \geq wt(b) = 3$ , if  $x_1^{(3)} \neq 0$ , then  $wt(x_1^{(3)}) = 3$ . Therefore, using the same method as in Lemma 3, we know that  $I_1 \leq p^{22}$  and  $I \leq 4p^{22}$ . Secondly, we compute the value of  $II$ . For  $\gamma_{x(3)} = (1, 1, 0, 0)$ , we have the following:

$$\begin{aligned} &\sum_{\gamma_{x(3)}=(1,1,0,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &= \sum_{i=2}^3 \sum_{j=2}^3 \sum_{wt(x_1^{(3)})=i, wt(x_2^{(3)})=j} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \end{aligned}$$

Since  $wt(x_1^{(3)}) = 2$ ,  $wt(x_2^{(3)}) = 2$ , by Theorem 2,

$$\begin{aligned} \max_{wt(x_1^{(3)})=2, wt(x_2^{(3)})=2} DP_3(a, x^{(3)}) &\leq p^{12} \sum_{j_1=3}^3 \sum_{j_2=3}^3 N[\gamma_{\pi(a)}, (j_1, j_2, 0, 0)] p^{8-j_1-j_2} \\ &= p^{12} \cdot N[(1, 1, 1, 0), (3, 3, 0, 0)] p^2 \\ &= p^{14}. \end{aligned}$$

Since  $wt(x_1^{(3)}) = 2, wt(x_2^{(3)}) = 2$  and  $\gamma_b = (1, 1, 0, 0)$ , the number of pattern whose form is  $(z_1^{(3)}, z_2^{(3)}, z_3^{(3)}, 0)$  is equal to  $N[(1, 1, 1, 0), (2, 2, 0, 0)] = 6$ . For each of  $(\gamma_1, \gamma_2, \gamma_3, 0)$ , by Lemma 2,

$$\begin{aligned} & \sum_{\gamma_{y_i^{(3)}}=\gamma_i, 1 \leq i \leq 3} DP_1(z^{(3)}, b) \\ &= \sum_{\gamma_{y_1^{(3)}}=\gamma_1} DP_1(z_1^{(3)}, b_1) \sum_{\gamma_{y_2^{(3)}}=\gamma_1} DP_1(z_2^{(3)}, b_2) \sum_{\gamma_{y_3^{(3)}}=\gamma_1} DP_1(z_3^{(3)}, b_3) \\ &\leq p^{12-(wt(\gamma_1)+wt(\gamma_2)+wt(\gamma_3))} = p^8. \end{aligned}$$

Therefore,  $\sum_{wt(x_1^{(3)})=2, wt(x_2^{(3)})=2} DP_1(z^{(3)}, b) \leq 6p^8$  and we arrive at the following:

$$\begin{aligned} & \sum_{wt(x_1^{(3)})=2, wt(x_2^{(3)})=2} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \\ &\leq \max_{wt(x_1^{(3)})=2, wt(x_2^{(3)})=2} DP_3(a, x^{(3)}) \sum_{wt(x_1^{(3)})=2, wt(x_2^{(3)})=2} DP_1(z^{(3)}, b) \\ &\leq 6p^{22}. \end{aligned}$$

Using the same method, we can have the followings:

$$\begin{aligned} & \sum_{wt(x_1^{(3)})=2, wt(x_2^{(3)})=3} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \leq 3(3p^{15} + p^{14})p^7, \\ & \sum_{wt(x_1^{(3)})=3, wt(x_2^{(3)})=2} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \leq 3(3p^{15} + p^{14})p^7, \\ & \sum_{wt(x_1^{(3)})=3, wt(x_2^{(3)})=3} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \leq (6p^{16} + 6p^{15} + p^{14})p^6. \end{aligned}$$

Therefore, we arrive at

$$\sum_{\gamma_{x^{(3)}}=(1,1,0,0)} DP_3(a, x^{(3)}) DP_1(z^{(3)}, b) \leq 6p^{22} + 6(3p^{15} + p^{14})p^7 + (6p^{16} + 6p^{15} + p^{14})p^6$$

and

$$II \leq 6[6p^{22} + 6(3p^{15} + p^{14})p^7 + (6p^{16} + 6p^{15} + p^{14})p^6],$$

because the upper bound on summation for distinct  $\gamma_{x^{(3)}}$  such that  $wt(\gamma_{x^{(3)}}) = 2$  is the same as the upper bound on summation for  $\gamma_{x^{(3)}} = (1, 1, 0, 0)$ . Using the same method, we have the following:

$$\begin{aligned} III &\leq 4[24p^{21} + 27(3p^{16} + p^{15})p^5 + 9(9p^{17} + 6p^{16} + p^{15})p^4 \\ &\quad + (24p^{18} + 27p^{17} + 9p^{16} + p^{15})p^3]. \end{aligned}$$



At last, the upper bound on  $IV$  can be computed by Theorem 1 as following:

$$IV \leq \max_{wt(x^{(3)})=4} DP_3(a, x^{(3)}) \leq \max_{wt(x^{(3)})=4, z^{(1)}} DP_2(z^{(1)}, x^{(3)}) \leq p^{16}.$$

Therefore, we arrive at

$$DP_4(a, b) = I + II + III + IV \leq 184p^{22} + 912p^{21} + 438p^{20} + 72p^{19} + 4p^{18} + p^{16}.$$

Therefore, the proof is completed.

Theorem 3 shows the upper bound on the maximal differential probability for 4 rounds of Rijndael-like structures and this is the main result of this paper.

**Theorem 3.**

$$DP_4(a, b) \leq \max\{4p^{19} + 6p^{18} + 4p^{17} + p^{16}, 184p^{22} + 912p^{21} + 438p^{20} + 72p^{19} + 4p^{18} + p^{16}\}.$$

*Proof.* We compute the upper bound on  $DP_4(a, b)$  for the value of  $wt(\gamma_{\pi(a)})$  and  $wt(b)$ . Since  $\beta_d = 5$ , if  $wt(\gamma_{\pi(a)}) + wt(b) \leq 4$ , then  $DP_4(a, b) = 0$ . Therefore, it is sufficient to compute the upper bound on  $DP_4(a, b)$ , when  $wt(\gamma_{\pi(a)}) + wt(b) \geq 5$ .

(i) If  $wt(\gamma_{\pi(a)}) = 4$ , then, by Theorem 1,

$$DP_4(a, b) = \sum_{x^{(2)}} DP_2(a, x^{(2)}) DP_2(z^{(2)}, b) \leq \max_{x^{(2)}} DP_2(a, x^{(2)}) \leq p^{16}.$$

(ii) If  $wt(b) = 4$ , then, by Theorem 1,

$$DP_4(a, b) = \sum_{x^{(2)}} DP_2(a, x^{(2)}) DP_2(z^{(2)}, b) \leq \max_{x^{(2)}} DP_2(a, x^{(2)}) \leq p^{16}.$$

(iii) If  $wt(\gamma_{\pi(a)}) = 2, wt(b) = 3$ , then, by Lemma 3,

$$DP_4(a, b) \leq 4p^{19} + 6p^{18} + 4p^{17} + p^{16}.$$

(iv) If  $wt(\gamma_{\pi(a)}) = 3, wt(b) = 2$ , then, by Lemma 4,

$$DP_4(a, b) \leq 4p^{19} + 6p^{18} + 4p^{17} + p^{16}.$$

(v) If  $wt(\gamma_{\pi(a)}) = 3, wt(b) = 3$ , then, by Lemma 5,

$$DP_4(a, b) \leq 184p^{22} + 912p^{21} + 438p^{20} + 72p^{19} + 4p^{18} + p^{16}.$$

When  $p \leq 2^{-3}$ , the maximum differential probability for 4 rounds of Rijndael-like structures is bounded by  $4p^{19} + 6p^{18} + 4p^{17} + p^{16}$ .

Using the similar method as in Theorem 3, we can compute the upper bound on the linear hull probability for 4 rounds of Rijndael-like structures.

**Theorem 4.**

$$LP_4(a, b) \leq \max\{4q^{19} + 6q^{18} + 4q^{17} + q^{16}, 184q^{22} + 912q^{21} + 438q^{20} + 72q^{19} + 4q^{18} + q^{16}\}.$$

We know that the differential probabilities for 5 rounds of Rijndael-like structures are smaller than or equal to the maximum differential probability for 4 rounds of Rijndael-like structures.

$$DP_5(a, b) = \sum_{x^{(4)}} DP_4(a, x^{(4)}) DP_1(z^{(4)}, b) \leq \max_{x^{(4)}} DP_4(a, x^{(4)}).$$

Similarly, we know that the differential probabilities for  $r$  ( $r \geq 5$ ) rounds of Rijndael-like structures are smaller than or equal to the maximum differential probability for 4 rounds of Rijndael-like structures. Therefore, the upper bound on the maximum differential probability and the linear hull probability for 4 rounds of Rijndael-like structures in Theorem 3 and Theorem 4 is the upper bound for  $r$  ( $r \geq 5$ ) rounds of Rijndael-like structures.

By applying our method to Rijndael, since  $p = q = 2^{-6}$  and  $\beta_d = \beta_l = 5$ , the upper bound on  $DP_4(a, b)$  and  $LP_4(a, b)$  is the following:

$$4 \times 2^{-114} + 6 \times 2^{-108} + 4 \times 2^{-102} + 2^{-96} \approx 1.06 \times 2^{-96}.$$

## 5 Conclusion

In this paper, we have proposed a new method for upper bounding the maximum differential probability and the maximum linear hull probability for Rijndael-like structures. We have proved that the maximum differential probability for 4 rounds of Rijndael-like structures is bounded by  $4p^{19} + 6p^{18} + 4p^{17} + p^{16}$ , when the maximum differential probability for S-boxes is  $p(\leq 2^{-3})$ . Also, we have proved that the maximum linear hull probability for 4 rounds of Rijndael-like structures is bounded by  $4q^{19} + 6q^{18} + 4q^{17} + q^{16}$ , when the maximum linear hull probability for S-boxes is  $q(\leq 2^{-3})$ . By applying our method to Rijndael, an improved upper bound  $1.06 \times 2^{-96}$  is obtained.

## References

1. Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In Douglas R. Stinson and Stafford Tavares, editors, *Selected Areas in Cryptography*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2000.
2. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystem. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - Crypto'90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer-Verlag, Berlin, 1991.

3. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
4. C.E.Shannon. Communication Theory of Secrecy System. *Bell System Technical Journal*, 28:656–715, October 1949.
5. Joan Daemen, René Govaerts, and Joos Vandwalle. Correlation matrices. In Bart Preneel, editor, *Fast Software Encryption, Second International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 275–285. Springer, 1994.
6. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.
7. Joan Daemen and Vincent Rijmen. Rijndael, AES Proposal. <http://www.nist.gov/aes>, 1998.
8. Seokhie Hong, Sangjin Lee, Jongin Lim, Jaechul Sung, Donghyeon Cheon, and Inho Cho. Provable security against differential and linear cryptanalysis for the spn structure. In Bruce Schneier, editor, *Fast Soft Encryption, 7th International Workshop*, pages 273–283, 2000.
9. Ju-Sung Kang, Seokhie Hong, Sangjin Lee, Okyeon Yi, Choonsik Park, and Jongin Lim. Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks. *ETRI Journal*, 23(4):158–167, 2001.
10. Liam Keliher, Henk Meijer, and Stafford Tavares. Improving the upper bound on the maximum average linear hull probability for rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography, 8th Annual International Workshop*, volume 2259 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2001.
11. Liam Keliher, Henk Meijer, and Stafford Tavares. New method for upper bounding the maximum average linear hull probability for spns. In Birgit Pfitzmann, editor, *Advances in Cryptology - Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 420–436. Springer-Verlag, Berlin, 2001.
12. Chae Hoon Lim. CRYPTON, AES Proposal. <http://www.nist.gov/aes>, 1998.
13. Chae Hoon Lim. A revised version of crypton - crypton v1.0 -. In Lars Knudsen, editor, *Fast Software Encryption, 6th International Workshop*, volume 1636 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 1999.
14. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseeth, editor, *Advances in Cryptology - Eurocrypt'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, Berlin, 1994.
15. NTT-Nippon Telegraph and Telephone Corporation. E2: Efficient Encryption algorithm, AES Proposal. <http://www.nist.gov/aes>, 1998.

# Threshold Cryptosystems Based on Factoring

Jonathan Katz<sup>1,3</sup> and Moti Yung<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Maryland (College Park),  
jkatz@cs.umd.edu

<sup>2</sup> Department of Computer Science, Columbia University,  
moti@cs.columbia.edu

<sup>3</sup> Work done while at Columbia University and Telcordia Technologies

**Abstract.** We consider threshold cryptosystems over a composite modulus  $N$  where the *factors* of  $N$  are shared among the participants as the secret key. This is a new paradigm for threshold cryptosystems based on a composite modulus, differing from the typical treatment of RSA-based systems where a “decryption exponent” is shared among the participants. Our approach yields solutions to some open problems in threshold cryptography; in particular, we obtain the following:

1. *Threshold Homomorphic Encryption.* A number of applications (e.g., electronic voting or efficient multi-party computation) require threshold homomorphic encryption schemes. We present a protocol for threshold decryption of the homomorphic Goldwasser-Micali encryption scheme [34], answering an open question of [21].
2. *Threshold Cryptosystems as Secure as Factoring.* We describe a threshold version of a variant of the signature standards ISO 9796-2 and PKCS#1 v1.5 (cf. [39, Section 11.3.4]), thus giving the first threshold signature scheme whose security (in the random oracle model) is equivalent to the hardness of factoring [12]. Our techniques may be adapted to distribute the Rabin encryption scheme [44] whose semantic security may be reduced to the hardness of factoring.
3. *Efficient Threshold Schemes without a Trusted Dealer.* Because our schemes only require sharing of  $N$  – which furthermore need not be a product of strong primes – our schemes are very efficient (compared to previous schemes) when a trusted dealer is not assumed and key generation is done in a distributed manner.

Extensions to achieve robustness and proactivation are also possible with our schemes.

## 1 Introduction

Threshold cryptosystems provide for increased security and availability of a particular cryptographic protocol by distributing the protocol among a number of participants. In a  $k$ -out-of- $\ell$  threshold scheme, the protocol is distributed in such a way that an adversary who corrupts at most  $k - 1$  participants (and learns all their local information) still cannot determine the secret key of the system or break the underlying cryptographic protocol. On the other hand, increased availability is achieved by ensuring that only  $k$  participants are needed in order

to carry out the computation and deliver the result. Going further, systems can be designed in a *robust* manner, such that even a malicious adversary who causes up to  $k - 1$  ( $k \leq \ell/2$ ) players to deviate arbitrarily from the protocol cannot prevent the correct output from being computed. Threshold schemes can also be *proactivized* to withstand the compromise of even all participants over the lifetime of the protocol, as long as only  $k - 1$  participants are corrupted during each time period; they may also be extended to handle *adaptive* adversaries who decide whom to corrupt at any point during execution of the protocol.

A long line of research has focused on threshold cryptography, with particular emphasis on threshold signature schemes (in many cases, deriving a threshold decryption scheme from a related signature scheme is easy). The approach was initiated by [17,18,19], and the first provably secure schemes for RSA- and discrete-logarithm-based signature schemes were given in [16,30,35]. Subsequent work focused on adding robustness to existing schemes [24,31,32] and on threshold decryption schemes with security against chosen-ciphertext attacks [47,9,20].

The above protocols are all proven secure with respect to a non-adaptive adversary who must choose which participants to corrupt before protocol execution begins (this is the type of adversary we consider here). Many recent works have dealt with stronger classes of adversaries, including adaptive [2,7] and proactive [41] adversaries. We refer the reader elsewhere for more comprehensive surveys of the existing literature (e.g., [28,36]).

The protocols mentioned above assume a dealer who distributes keys to the participants before the protocol begins. The dealer must be minimally trusted not to reveal the secret key and therefore represents a single point of failure for the entire system. Thus, it is often desirable to distribute the key-generation phase among the participants. This was first accomplished for discrete-logarithm-based cryptosystems in [32,8] (building on [43]), and for RSA-based cryptosystems in [5] (for passive adversaries) and [27] (for active adversaries).

There is still a need to design threshold schemes for many *specific* cryptosystems (most previous research on threshold cryptography was restricted to RSA- and discrete-logarithm-based schemes). First, note that for threshold cryptography to become truly practical, it remains important to improve the efficiency and conceptual simplicity of existing solutions.<sup>1</sup> Furthermore, as pointed out many times previously [29,21,14,37,13], threshold *homomorphic* encryption schemes are useful for achieving such goals as electronic voting and efficient multi-party computation. Threshold schemes have been given previously [43,21,14] for the El Gamal (which is homomorphic under group multiplication) and Paillier [42] (which is homomorphic under addition) cryptosystems. Yet, for some applications, homomorphism over, e.g.,  $\mathbb{Z}_2$  is required or sufficient [29,38,37,13] and hence other homomorphic schemes may not work or may be “overkill” for the problem at hand. Clearly, additional approaches yielding threshold homomorphic encryption are needed (and this was left as an explicit open question in [21]; see also [13]).

<sup>1</sup> This is the motivation for the study of threshold cryptography since, in a theoretical sense, “solutions” already exist using generic multi-party computation [33].

## 1.1 Our Contributions

**THRESHOLD HOMOMORPHIC ENCRYPTION.** We show how to achieve threshold decryption for the Goldwasser-Micali (GM) encryption scheme [34], whose security is based on the hardness of deciding quadratic residuosity. The GM encryption scheme is homomorphic over  $\mathbb{Z}_2$ . As mentioned above, (semantically-secure) threshold homomorphic encryption schemes have many important applications; for example, efficient multi-party computation can be based on any (efficient) scheme of this type [29,13]. Threshold GM encryption can also be used for distributed tallying in electronic voting [37].

Concurrent with the present work, a variant threshold GM-like cryptosystem has been constructed [13] using an alternate approach. However, this scheme (which builds on [29]) requires the DDH assumption in  $\mathbb{Z}_N^*$ , whereas the security of our construction relies only on the quadratic residuosity assumption. Indeed, eliminating this assumption is left as an open question in [13]. We believe our solution also offers a more efficient and conceptually simpler method. Finally, our scheme has the added advantage of allowing for efficient distributed key generation when a trusted dealer is not assumed; this is not possible in [13] because they require  $N$  to be a product of safe primes<sup>2</sup>.

**THRESHOLD CRYPTOSYSTEMS BASED ON FACTORING.** We are not aware of any previous constructions of threshold cryptosystems whose security can be reduced to the assumption that factoring is hard. Here, we propose a novel and efficient distributed version of the Rabin-Williams signature scheme [39, Section 11.3.4] (see also [44]), variants of which have been standardized. Security of this scheme has recently been shown [12] to be equivalent to the hardness of factoring in the random oracle model (see also earlier work of [3]).

**EFFICIENCY IMPROVEMENTS.** The protocols we present are all efficient and *practical* threshold schemes. When a trusted dealer cannot be assumed (and key generation must therefore be done in a distributed fashion), our threshold schemes are more efficient than previous solutions not requiring a trusted dealer [15,22]. The threshold schemes presented here may be easily executed in a modular manner following a “streamlined” version of the distributed key-generation protocols of [5,27]: all information required by the present schemes is in place upon completion of these key-generation protocols, and we do not require that  $N$  be a product of safe primes. A “streamlined” version of these protocols may be used because *we do not require computation of an inverse over a shared (secret) modulus* (and therefore are done once  $N$  has been generated). We are therefore able to avoid *altogether* the step whose efficiency is improved by [10].

Finally, we believe the methods outlined in this paper are interesting in their own right; the sharing of the factors of  $N$  alone, without the need to additionally share a “decryption exponent”, is a new paradigm for threshold cryptography over composite moduli and may prove useful in the design of future schemes. It

<sup>2</sup> The recent work of [1] shows how  $N$  of this form can be generated efficiently in a distributed fashion; even so, it remains more efficient to generate  $N$  without this added requirement.

is specifically useful whenever the function to be computed can be expressed as a combination of the factors and where the computation of its partial results is enabled by shares of the factors.

## 2 Model and Definitions

### 2.1 The Model

**PARTICIPANTS.** The participants are  $\ell$  servers  $\{P_1, \dots, P_\ell\}$  and a trusted dealer  $D$ <sup>3</sup>. The dealer generates a public key  $N$  for the underlying cryptosystem and distributes shares to each of the participants. After the dealing phase, the dealer does not take part in executions of the protocol. Following [30], we assume the participants are connected by a complete network of private channels. In addition, all players have access to an authenticated broadcast channel so that the true sender of a message can always be correctly determined. These assumptions allow us to focus on high-level descriptions of the protocols; however, they may be instantiated using standard cryptographic techniques (in the proactive setting, care needs to be taken; see [41,35]).

**THE ADVERSARY.** Our  $k$ -out-of- $\ell$  schemes assume a non-adaptive adversary who may corrupt up to  $k - 1$  participants in advance of protocol execution. The adversary has access to all information available to the corrupted players, including their secret keys, messages they receive, and messages broadcast to all players. One may consider two types of adversaries: *passive* adversaries who follow the protocol faithfully yet monitor all information available to corrupted participants, and *active* adversaries who may cause participants to deviate arbitrarily from the protocol. We consider both types of adversaries in what follows. In the case of threshold signature schemes, the adversary may submit signing requests to the system at any time; in the case of threshold decryption, we consider both chosen plaintext and chosen ciphertext attacks.

### 2.2 Security

Formal definitions of security for threshold cryptosystems have appeared elsewhere [31]. We describe, informally, our requirements. First, we want the security of the threshold scheme to be equivalent to the security of the original scheme even when an adversary has corrupted  $k - 1$  servers and obtained all their local information. To prove that this requirement is met, we reduce the security of the threshold scheme to that of the original scheme by showing how an adversary attacking the original scheme can simulate the view of (up to)  $k - 1$  servers in the threshold scheme. Following [31], we call such threshold protocols *simulatable*. An additional requirement we will consider is *robustness*: for any active adversary who causes at most  $k - 1$  ( $k \leq \ell/2$ ) participants to deviate arbitrarily from the protocol, the correct result can always be computed by the remaining (uncorrupted) participants.

<sup>3</sup> We stress that this trusted dealer is not essential to our schemes since a distributed algorithm (adapting [5,27]) may be run when a dealer is not available.

### 3 A Threshold Homomorphic Encryption Scheme

We begin by describing how to achieve threshold decryption for the well-known homomorphic encryption scheme of Goldwasser and Micali [34] (henceforth, GM). The GM encryption scheme is as follows: the public key is a composite  $N = pq$ , where  $p$  and  $q$  are prime and  $p = q = 3 \bmod 4$ . The private key consists of the factorization of  $N$ . To encrypt bit  $b \in \{0, 1\}$ , choose a random element  $r \in \mathbb{Z}_N$  and send  $C = (-1)^b r^2 \bmod N$ . Decryption of ciphertext  $C$  proceeds by determining whether  $C$  is a quadratic residue or not. To do this, first calculate the Jacobi symbol  $J = (\frac{C}{N})$ . If  $J \neq 1$ , the ciphertext is ill-formed (i.e., the encryption algorithm was not run honestly, or else the message was corrupted in transmission); therefore, simply output  $\perp$ . If  $J = 1$ , we may decide whether  $C$  is a quadratic residue by computing  $b' = C^{(N-p-q+1)/4} \bmod N$ ; note that  $b' = \pm 1$  and furthermore  $C$  is a quadratic residue iff  $b' = 1$ . The original plaintext can be recovered as  $b = (1 - b')/2$ . This scheme is semantically secure under the quadratic residuosity assumption [34].

#### 3.1 An $\ell$ -out-of- $\ell$ Protocol

For simplicity and clarity of exposition, we describe in this section a protocol for “basic” threshold GM decryption (cf. Figure 1) which assumes a trusted dealer and is an  $\ell$ -out-of- $\ell$  solution. Thus, all  $\ell$  participants are needed in order to decrypt a ciphertext; on the other hand, it remains infeasible for any adversary who corrupts  $\ell - 1$  or fewer participants to decrypt a given ciphertext. In the following section, we discuss extensions and modifications which allow for the

##### Dealing Phase

Input: Composite  $N$  and primes  $p, q$  ( $|p| = |q| = n$ ) such that  $N = pq$  with  $p, q = 3 \bmod 4$

1. Choose  $p_1, q_1, \dots, p_\ell, q_\ell \in_R (0, 2^{2n})$  such that  $p_i = q_i = 0 \bmod 4$ , for all  $i$
2. Set  $p_0 = p - \sum_{i=1}^{\ell} p_i$  and  $q_0 = q - \sum_{i=1}^{\ell} q_i$
3. Send  $(p_i, q_i)$  to player  $i$
4. Broadcast  $(N, p_0, q_0)$

##### Decryption Phase

Input: Ciphertext  $C$

1. All players compute  $J = (\frac{C}{N})$  (this computation is done publicly)
2. If  $J \neq 1$ , all players output  $\perp$  and stop
3. Otherwise ( $J = 1$ ), player  $i$  broadcasts  $b_i = C^{(-p_i - q_i)/4} \bmod N$
4. All players publicly compute  $b_0 = C^{(N - p_0 - q_0 + 1)/4} \bmod N$
5. The decrypted bit  $b$  is computed as  $b = (1 - \prod_{i=0}^{\ell} b_i \bmod N) / 2$

**Fig. 1.**  $\ell$ -out-of- $\ell$  decryption for the GM cryptosystem



more general  $k$ -out-of- $\ell$  threshold, provide robustness, and enable proactivation of the protocol. Additionally, we discuss how to remove the trusted dealer and perform the initial key generation in a distributed manner.

**KEY DISTRIBUTION.** The dealer generates primes  $p, q \equiv 3 \pmod{4}$  (where  $|p| = |q| = n$ ) and sets  $N = pq$ . The public key is  $N$ , and the private key is computed as  $d = (N - p - q + 1)/4$ ; note that  $d$  is always an integer. For all  $i$ , the dealer chooses integers  $p_i, q_i \in_R (0, 2^{2n})$  such that  $p_i \equiv q_i \equiv 0 \pmod{4}$ . Finally, the dealer sets  $p_0 = p - \sum_{i=1}^{\ell} p_i$  and  $q_0 = q - \sum_{i=1}^{\ell} q_i$ . The dealer sends  $(p_i, q_i)$  to player  $i$  and broadcasts  $(N, p_0, q_0)$ . We note that it would suffice for the dealer to send  $(p_i + q_i)/4$  to each party – and this is likely what would be done in practice – but we prefer the present description for pedagogical reasons.

**DECRYPTION.** Decryption of a ciphertext  $C$  proceeds as follows: first, the Jacobi symbol  $J = (\frac{C}{N})$  is computed; this can be computed in polynomial time even without knowledge of the factorization of  $N$ . If  $J \neq 1$ , all players simply output  $\perp$ . Otherwise, player  $i$  outputs  $b_i = C^{(-p_i - q_i)/4} \pmod{N}$  (note that, by design, the exponent is an integer and hence  $b_i$  can be efficiently computed). Players publicly compute  $b_0 = C^{(N - p_0 - q_0 + 1)/4} \pmod{N}$  (again, by design, the exponent is an integer). Deciding whether  $C$  is a quadratic residue may be done by computing  $b' = \prod_{i=0}^{\ell} b_i \pmod{N}$ . The decrypted bit is simply  $b = \frac{1 - b'}{2}$ .

**Theorem 1.** *The protocol of Figure 1 is simulatable for any adversary who passively eavesdrops on at most  $\ell - 1$  parties. This implies the semantic security of the encryption scheme for such an adversary, assuming the hardness of deciding quadratic residuosity.*

The proof is similar to the more involved proof of security for the Rabin-Williams signature scheme given below (cf. Theorem 4), and is therefore omitted.

### 3.2 Extensions

**REDUCING THE THRESHOLD.** It is a severe limitation to require  $\ell$  active servers in order to decrypt. More preferable is a  $k$ -out-of- $\ell$  solution in which only  $k$  servers are required for decryption. A number of techniques exist for accomplishing this using the above protocol as a starting point; we sketch two such solutions here (but see [4] for another approach).

One approach is to adapt the suggestions of Rabin [45] to our setting. First, the dealer fixes a prime  $P > 2^{2n}$  which is broadcast to all participants. Then, for each  $p_i$  (and also  $q_i$ ), the dealer chooses a random  $(k - 1)$ -degree polynomial  $f_i(\cdot)$  over the field  $\mathbb{Z}_P$  such that  $f_i(0) = p_i$ . To player  $j$ , the dealer sends  $f_i(j)$  for  $1 \leq i \leq \ell$ . This achieves a  $k$ -out-of- $\ell$  secret sharing of the  $\{p_i\}$  (and also the  $\{q_i\}$ ). Decryption proceeds as before, with each player  $i$  broadcasting its share  $b_i$ . In addition, players prove correctness of their shares using one of the robustness techniques described below. If player  $i$  cannot prove correctness of his share (or, more generally, if player  $i$  fails to participate), the remaining players can publicly reconstruct  $(p_i, q_i)$  using the shares they have been given. The correct share  $b_i$  may then be computed publicly and included in the calculation of  $b$ . We note

that, in case a trusted dealer is not available, each player may itself deal shares of  $(p_i, q_i)$  to the other players. If robustness is desired for this step, verifiable secret sharing (VSS) may be used. Details appear in [45].

A problem with this approach is that it may unfairly penalize servers which are temporarily off-line or otherwise unable to participate in an execution of the protocol. If this happens, this player's share is publicly reconstructed and hence available to an adversary eavesdropping on the protocol. Note that it may be much easier for an adversary to disconnect or prevent communication from a player than to corrupt a player (even passively). By "disconnecting" users one-by-one – possibly in parallel – an adversary may be able to obtain the secret key of the system.<sup>4</sup>

An alternative is to use ideas motivated by the protocols of Frankel, et al. [26]. Let  $L = \ell!$ . Instead of the  $\ell$ -out-of- $\ell$  *additive* sharing illustrated in Figure 1, the dealer now performs  $k$ -out-of- $\ell$  *polynomial* sharing as follows: The dealer chooses  $s^* \in_R (0, 2^{2n})$  subject to  $s^* = 0 \bmod 4$ , and additionally chooses a  $(k-1)$ -degree polynomial  $f$  over the integers – with coefficients chosen uniformly from  $\{0, 4L, \dots, L^3 2^{3n} k\}$  – such that  $f(0) = L^2 s^*$ . The dealer distributes  $s_i \stackrel{\text{def}}{=} f(i)$  to player  $i$ . Finally, the dealer broadcasts the value  $p + q - L^2 s^*$ . To decrypt, the players first choose a random subset  $A$  consisting of  $k$  players. Each player in  $A$  computes the appropriate Lagrange interpolation coefficient  $z_{i,A}$  and sets his (temporary) share to  $\hat{s}_i = z_{i,A} \cdot s_i$ . Note that, due to the careful choice of the polynomial  $f$ , the  $\{\hat{s}_i\}$  may be computed over the integers and furthermore  $\hat{s}_i = 0 \bmod 4$  for all  $i$ . The  $\{\hat{s}_i\}_{i \in A}$  thus constitute a  $k$ -out-of- $k$  *additive* sharing of  $L^2 s^*$ , and may be used to decrypt as in Figure 1. Techniques to achieve robustness for the above approach are given in [26].

**Theorem 2.** *The protocol of Figure 1 modified using either of the approaches described above gives a  $k$ -out-of- $\ell$  protocol which is simulatable for any adversary who passively eavesdrops on at most  $k-1$  parties.*

**(Informal Idea of the) Proof.** The approach of Rabin [45] may be viewed as a “generic” approach which converts *any*  $\ell$ -out-of- $\ell$  scheme to a  $k$ -out-of- $\ell$  scheme. The approach of Frankel, et al. [26] must be more carefully modified for the cryptosystem at hand; for the modification sketched above, however, a proof follows easily using their techniques. ■

**ROBUSTNESS.** We may distinguish two methods for adding robustness to the above protocol: methods which work for arbitrary  $N$ , and methods which work only when  $N$  is a product of strong primes<sup>5</sup>. Methods specialized for the latter case can be more efficient; on the other hand, when distributed key generation is required, methods which work for arbitrary  $N$  may be preferred because distributed generation of  $N$  a product of safe primes [1] is less efficient.

Gennaro, et al. [31] give two methods for verifying correctness of the partial outputs  $b_i$  when  $N$  is a product of strong primes. One method, which is non-interactive, requires the dealer to distribute verification information to all players

<sup>4</sup> This was pointed out to us by an anonymous referee.

<sup>5</sup> That is,  $N = pq$  with  $p = 2p' + 1$  and  $q = 2q' + 1$ , where  $p, q, p', q'$  are all prime.

during the dealing phase; namely,  $V_{i,j}$  is sent to player  $i$  to enable his verification of player  $j$ . When executing the protocol, player  $i$  outputs  $b_i$  and also  $b_{i,j}$  for all  $j$ ; player  $j$  verifies the correctness of  $b_i$  using  $V_{j,i}$  and  $b_{i,j}$ . This requires  $O(\ell^2)$  memory for each player, and also increases the communication of the protocol (per player) to  $O(\ell^2)$ .

A second approach of [31] requires the dealer to choose a random element (of high order)  $g \in \mathbb{Z}_N^*$  and broadcast  $g$  along with *witnesses*  $w_i = g^{(-p_i - q_i)/4} \bmod N$ , for all  $i$ . After player  $i$  broadcasts  $b_i$ , he engages in an (interactive) zero-knowledge proof with all other players in which he proves that  $\log_g w_i = \log_C b_i$ . Unfortunately, this approach seems to require interaction even in the random oracle model. More recently, Shoup [46] (based on earlier work of [11]) describes a non-interactive, zero-knowledge proof (in the random oracle model) for equality of discrete logarithms. Here, players work in the subgroup of quadratic residues  $Q_N \subset \mathbb{Z}_N^*$ : the dealer chooses  $g \in Q_N$  and player  $i$  now proves that  $\log_g w_i = \log_{C^2} b_i^2$  (squaring is necessary to ensure that values are in  $Q_N$ ).

The above approaches suffice for  $N$  a product of strong primes. For general  $N$ , however, we must use other techniques to achieve robustness.<sup>6</sup> One possibility is to use the cryptographic program-checking method of [24], which requires interaction between each pair of parties (this interaction can be reduced to only two rounds using a random oracle). Another approach extends the witness-based approach above. Using a random oracle, players may, as above, give an efficient, non-interactive, zero-knowledge proof [11] that  $\log_g w_i = \log_C b_i$ . A difficulty here is that soundness is only guaranteed if  $g$  is of high order; however, as shown in [27], a set (of super-logarithmic size) of random elements of  $\mathbb{Z}_N^*$  generates a large-order subgroup of  $\mathbb{Z}_N^*$  with all but negligible probability. Soundness can thus be guaranteed by fixing such a set as part of the dealing phase and having players give a non-interactive proof with respect to each element in this set. Fouque and Stern [22] suggest another method for achieving robustness; they require  $N$  of a special form but show how such  $N$  can be generated efficiently in a distributed manner.

The above approaches to proving correctness of exponentiation modulo  $N$  allow proofs of correctness for the partial shares  $b_i$  broadcast by each player in the protocol. Theorems 1 and 2, together with the results cited above, thus yield the following theorem:

**Theorem 3.** *The protocol of Figure 1 augmented with any of the robustness techniques described above (appropriate for the modulus  $N$ ) and any of the approaches for achieving a  $k$ -out-of- $\ell$  ( $k \leq \ell/2$ ) threshold (as described in Theorem 2) results in a robust protocol which is simulatable for any adversary who actively controls at most  $k - 1$  parties.*

**REMOVING THE TRUSTED DEALER.** The efficiency improvement of the current protocol is most evident when a trusted dealer is not assumed, and the public modulus must be generated in a distributed fashion. In this case, our scheme has

<sup>6</sup> Although we still refer to a dealer, the techniques described here can be implemented easily following the (robust) distributed key-generation protocol of [27].

two advantages: (1) moduli of a special form (i.e.,  $N$  a product of strong primes) are not required, in contrast with some recent solutions (e.g., [46]). (Even though a protocol has recently been given [1] for efficiently generating  $N$  of this form in a distributed fashion, this protocol remains less efficient than protocols for more general  $N$  [5,27].) Furthermore, (2) an expensive step of the distributed key-generation protocol can be skipped entirely. Specifically, computation of an inverse<sup>7</sup> over  $\varphi(N)$  (recall that  $\varphi(N)$  must remain hidden from the players) is not required in our scheme.

The protocol of Figure 1 may be combined modularly with the distributed key-generation protocols of [5,27]. Following execution of these key-generation protocols, all the players *already* have additive shares  $(p_i, q_i)$  of the factors of  $N$ . A small complication is that the protocol requires all players to have  $p_i = q_i = 0 \bmod 4$ . To deal with this, simply have player  $i$  choose  $p_i = q_i = 0 \bmod 4$ . Additionally, the “public remainder” may be set to  $(p_0, q_0) = (3, 3)$ . Decryption is then done as before. A similar approach was used in, e.g., [5] where they require  $p = q = 3 \bmod 4$ .

**PROACTIVE SECURITY.** Proactive security may be added to our protocols using known techniques. For example, if the approach of Rabin [45] is used to achieve  $k$ -out-of- $\ell$  threshold, the generic proactivation techniques given there will work here as well. Similarly, if the approach of Frankel, et al. [26] is used, the proactivation techniques given there will also work for the present protocol. Due to space limitations, we refrain from a detailed description of these techniques.

**CHOSEN-CIPHERTEXT SECURITY.** A generic method for making threshold cryptosystems secure against chosen-ciphertext attack was recently described [20], adapting the method of Naor and Yung [40] for the random oracle model. What is required are two schemes and an honest-verifier ZK proof of knowledge that two encryptions are of the same plaintext. Such a proof system for the GM cryptosystem is presented in Appendix A. Although the protocol given there is interactive, it can be made non-interactive (and reasonably efficient) in the random-oracle model.

## 4 A Threshold Signature Scheme Based on Factoring

Distributing the prime factors of the modulus among the participants offers a new paradigm for the construction of threshold systems over composite moduli. As a further example of the applicability of our technique, we describe a method for distributing the Rabin-Williams signature scheme [44], variants of which have been standardized as ISO 9796-2 and PKCS#1 v1.5. This scheme is particularly interesting since it offers the first threshold signature scheme whose security can be based on the hardness of factoring (in the random oracle model) [12].

---

<sup>7</sup> This is precisely the step whose efficiency is improved by [10]. Here, we avoid this step altogether!

#### 4.1 The (Modified) Rabin Signature Scheme

The modified Rabin signature scheme [39, Section 11.3.4] is defined as follows: a public key is generated by choosing two primes  $p, q$  of length  $n$  such that  $p \equiv 3 \pmod{8}$  and  $q \equiv 7 \pmod{8}$ . The public key is set to  $N = pq$  ( $N$  of this form are called *Williams integers*). The private key is  $d = (N - p - q + 5)/8$ .

Messages  $m$  to be signed are assumed to be appropriately encoded and the resulting underlying message space is  $\mathcal{M} = \{m : m \equiv 6 \pmod{16}\}$  (see [12]). First, the Jacobi symbol  $J = (\frac{m}{N})$  is computed. If  $J = 1$ , set  $\tilde{m} = m$ ; if  $J = -1$ , set  $\tilde{m} = m/2$  (note that there is only negligible probability that  $J \neq 1, -1$ ). The signature is computed as  $s = \tilde{m}^d \pmod{N}$ .

To verify signature  $s$  on message  $m$  (where  $m \equiv 6 \pmod{16}$ ), first compute  $\tilde{m} = s^2 \pmod{N}$ . Then, verify the following:

- If  $\tilde{m} \equiv 6 \pmod{8}$ , verify whether  $m \stackrel{?}{=} \tilde{m}$
- If  $\tilde{m} \equiv 3 \pmod{8}$ , verify whether  $m \stackrel{?}{=} 2\tilde{m}$
- If  $\tilde{m} \equiv 7 \pmod{8}$ , verify whether  $m \stackrel{?}{=} N - \tilde{m}$
- If  $\tilde{m} \equiv 2 \pmod{8}$ , verify whether  $m \stackrel{?}{=} 2(N - \tilde{m})$

We refer the reader to [39, Section 11.3.4] for a proof of correctness and further discussion.

#### 4.2 An $\ell$ -out-of- $\ell$ Protocol

As above, we present the  $\ell$ -out-of- $\ell$  solution here for simplicity (cf. Figure 2); extensions as discussed in Section 3.2 are applicable here as well.

##### Dealing Phase

Input: Composite  $N$  and primes  $p, q$  ( $|p| = |q| = n$ ) such that  $N = pq$   
with  $p \equiv 3 \pmod{8}$  and  $q \equiv 7 \pmod{8}$

1. Choose  $p_1, q_1, \dots, p_\ell, q_\ell \in_R (0, 2^{2n})$  such that  $p_i \equiv q_i \equiv 0 \pmod{8}$ , for all  $i$
2. Set  $p_0 = p - \sum_{i=1}^{\ell} p_i$  and  $q_0 = q - \sum_{i=1}^{\ell} q_i$
3. Send  $(p_i, q_i)$  to player  $i$
4. Broadcast  $(N, p_0, q_0)$

##### Signature Generation Phase

Input: Message  $m \equiv 6 \pmod{16}$  (appropriately encoded)

1. Player  $i$  computes  $J = (\frac{m}{N})$  (this computation is done publicly)
2. If  $J = 1$ , set  $\tilde{m} = m$ ; else set  $\tilde{m} = m/2$
3. Player  $i$  broadcasts  $s_i = \tilde{m}^{(-p_i - q_i)/8} \pmod{N}$
4. All players publicly compute  $s_0 = \tilde{m}^{(N - p_0 - q_0 + 5)/8} \pmod{N}$
5. The signature  $s$  is computed as  $s = \prod_{i=0}^{\ell} s_i \pmod{N}$

**Fig. 2.**  $\ell$ -out-of- $\ell$  signing for the Rabin signature scheme

**KEY DISTRIBUTION.** The dealer generates primes  $p, q$  (where  $|p| = |q| = n$ ,  $p \equiv 3 \pmod{8}$ , and  $q \equiv 7 \pmod{8}$ ) and sets  $N = pq$ . The public key of the protocol is  $N$ , and the private key (see Section 4.1) is  $d = (N - p - q + 5)/8$ . For  $i = 1, \dots, \ell$ , the dealer then chooses  $p'_i, q'_i \in_R (0, 2^{2n})$  such that  $p_i = q_i = 0 \pmod{8}$ . The dealer sets  $p_0 = p - \sum_{i=1}^{\ell} p_i$  and  $q_0 = q - \sum_{i=1}^{\ell} p_i$ . Finally, the dealer sends  $(p_i, q_i)$  to player  $i$  and broadcasts  $(p_0, q_0)$ .

**SIGNATURE GENERATION.** We assume the message  $m \in \mathcal{M}$  to be signed is already encoded in some appropriate agreed-upon manner (i.e., as discussed above). First, the Jacobi symbol  $J = (\frac{m}{N})$  is computed publicly (note that the Jacobi symbol can be computed in polynomial time even without knowledge of the factorization of  $N$ ). If  $J = 1$ , define  $\tilde{m} = m$ ; if  $J = -1$ , define  $\tilde{m} = m/2$ ; this step may be done publicly as well.

The desired signature is  $s = \tilde{m}^d = \tilde{m}^{(N-p-q+5)/8} \pmod{N}$ . Player  $i$  broadcasts the value  $s_i = \tilde{m}^{(-p_i-q_i)/8} \pmod{N}$  (note that, by design, the exponent is an integer and hence  $s_i$  can be efficiently computed). Players publicly compute  $s_0 = \tilde{m}^{(N-p_0-q_0+5)/8} \pmod{N}$  (again, by design, the exponent is an integer). Finally, the signature is computed as  $s = \prod_{i=0}^{\ell} s_i \pmod{N}$ . Verification of the signature is exactly as described in Section 4.1.

**Theorem 4.** *The protocol of Figure 2 is simulatable for any adversary who passively eavesdrops on at most  $\ell - 1$  parties. This implies that the signature scheme is existentially unforgeable under chosen message attacks, assuming the hardness of factoring (in the random oracle model).*

*Proof.* A description of a simulator for the dealing phase and the signature generation phase appears in Figure 3. We assume (without loss of generality)

#### Simulation of Dealing Phase

Input: Composite  $N$  where  $|N| = 2n$

1. Choose  $p_1, q_1, \dots, p_{\ell}, q_{\ell} \in_R (0, 2^{2n})$  such that  $p_i = q_i = 0 \pmod{8}$
2. Choose random  $p^*, q^*$  such that  $|p^*| = |q^*| = n$ ,  $p^* \equiv 3 \pmod{8}$ , and  $q^* \equiv 7 \pmod{8}$
3. Set  $p_0 = p^* - \sum_{i=1}^{\ell} p_i$  and  $q_0 = q^* - \sum_{i=1}^{\ell} q_i$
4. Send  $(p_i, q_i)$  to player  $i$ , for  $1 \leq i \leq \ell - 1$
5. Broadcast  $(p_0, q_0)$

#### Simulation of Player $\ell$ in Signature Generation Phase

Input: Message  $m \equiv 6 \pmod{16}$  (appropriately encoded); signature  $s$

1. Compute  $J = (\frac{m}{N})$
2. If  $J = 1$ , set  $\tilde{m} = m$ ; else set  $\tilde{m} = m/2$
3. Compute  $s_i = \tilde{m}^{(-p_i-q_i)/8} \pmod{N}$ , for  $1 \leq i \leq \ell - 1$
4. Compute  $s_0 = \tilde{m}^{(N-p_0-q_0+5)/8} \pmod{N}$
5. Broadcast  $s_{\ell} = s / (\prod_{i=0}^{\ell-1} s_i) \pmod{N}$

**Fig. 3.** Simulator for  $\ell$ -out-of- $\ell$  threshold Rabin signature scheme

that the adversary eavesdrops on players  $1, \dots, \ell-1$ . Simulatability of the dealing phase is evident from the following:

- The  $\{p_i, q_i\}_{1 \leq i \leq \ell-1}$  have the same distribution as in a real execution of the protocol.
- The distribution on  $(p_0, q_0)$ , conditioned on the values of  $\{p_i, q_i\}_{1 \leq i \leq \ell-1}$  seen by the adversary, is statistically indistinguishable from the distribution on  $(p_0, q_0)$  in a real execution of the protocol. This is because, for *any*  $p, p^* < 2^{n+1}$ , the distributions  $\{p - p_1\}_{p_1 \in_R(0, 2^{2n})}$  and  $\{p^* - p_1\}_{p_1 \in_R(0, 2^{2n})}$  are statistically indistinguishable.

Simulatability of the signature generation phase (in particular, the value  $s_\ell$ ) follows easily from the simulatability of the dealing phase. ■

Efficient extensions to achieve optimal threshold, robustness, proactivation, and distributed key generation are all possible as outlined in Section 3.2. Also, the above method extends to give threshold decryption of the Rabin *encryption* scheme [44], whose semantic security may be based on the hardness of factoring.

## References

1. J. Algesheimer, J. Camenisch, and V. Shoup. Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products. Crypto 2002.
2. D. Beaver and S. Haber. Cryptographic Protocols Provably Secure Against Dynamic Adversaries. Eurocrypt '92.
3. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. Eurocrypt '96.
4. S.R. Blackburn. Combinatorics and Threshold Cryptography. In *Combinatorial Designs and their Applications*, F.C. Holroyd, et al., eds., CRC Press, 1999.
5. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. Crypto '97.
6. C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, eds., *Cryptography and Coding*, Clarendon Press, 1989.
7. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Multi-Party Computation. STOC '96.
8. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive Security for Threshold Cryptosystems. Crypto '99.
9. R. Canetti and S. Goldwasser. An Efficient Threshold Public-Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. Eurocrypt '99.
10. D. Catalano, R. Gennaro, and S. Halevi. Computing Inverses over a Shared Secret Modulus. Eurocrypt 2000.
11. D. Chaum and T. Pedersen. Wallet Databases and Observers. Crypto '92.
12. J.S. Coron. Security Proof for Partial-Domain Hash Signature Schemes. Crypto 2002.
13. R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. Eurocrypt 2001.
14. I. Damgård and M. Jurik. A Generalization, a Simplification, and Some Applications of Paillier's Probabilistic Public-Key System. PKC 2001.

15. I. Damgård and M. Koprowski. Practical Threshold RSA Signatures without a Trusted Dealer. Eurocrypt 2001.
16. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC '94.
17. Y. Desmedt. Society and Group-Oriented Cryptography: A New Concept. Crypto '87.
18. Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89.
19. Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signatures. Crypto '91.
20. P.-A. Fouque, and D. Pointcheval, Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. Asiacrypt 2001.
21. P.-A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. Financial Cryptography, 2000.
22. P.-A. Fouque and J. Stern. Fully Distributed Threshold RSA under Standard Assumptions. Asiacrypt 2001.
23. Y. Frankel. A Practical Protocol for Large Group-Oriented Networks. Eurocrypt '89.
24. Y. Frankel, P. Gemmell, and M. Yung. Witness-Based Cryptographic Program Checking and Robust Function Sharing. STOC '96.
25. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. Crypto '97.
26. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptography. FOCS '97.
27. Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA Key Generation. STOC '98.
28. Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-Secure Distributed Public-Key Systems. European Symposium on Algorithms '99.
29. M. Franklin and S. Haber. Joint Encryption and Message-Efficient Secure Computation. J. Crypto 9(4): 217–232 (1996).
30. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. Eurocrypt '96.
31. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. J. Crypto 13(2): 273–300 (2000).
32. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log-Based Cryptosystems. Eurocrypt '99.
33. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. STOC '87.
34. S. Goldwasser and S. Micali. Probabilistic Encryption. JCSS 28(2): 270–299 (1984).
35. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. CCCS '97.
36. S. Jarecki and A. Lysyanskaya, Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. Eurocrypt 2000.
37. J. Katz, S. Myers, and R. Ostrovsky. Cryptographic Counters and Applications to Electronic Voting. Eurocrypt 2001.
38. E. Kushilevitz and R. Ostrovsky. Replication is not Needed: Single Database Computationally-Private Information Retrieval. FOCS '97.
39. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1999.
40. M. Naor and M. Yung. Public-key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. STOC '90.
41. R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91.



42. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt '99.
43. T. P. Pedersen. A Threshold Cryptosystem Without a Trusted Party. Eurocrypt '91.
44. M. O. Rabin. Digital Signatures and Public Key Functions as Intractable as Factoring. Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.
45. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. Crypto '98.
46. V. Shoup. Practical Threshold Signatures. Eurocrypt 2000.
47. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Against Chosen Ciphertext Attack. Eurocrypt '98.

## A Proof of Equality for GM Ciphertexts

Input: Blum integers  $N_1, N_2$  and  $X_1, X_2$  where:  
 $\{X_1 = (-1)^b x_1^2 \bmod N_1, X_2 = (-1)^b x_2^2 \bmod N_2\}$  with  $x_j \in Z_{N_j}^*$  and  $b \in \{0, 1\}$ .

Repeat  $k$  times:

1. The prover chooses a random bit  $c$  and “twin encrypts” it; i.e.,  
 $\{V_1 = (-1)^c v_1^2 \bmod N_1, V_2 = (-1)^c v_2^2 \bmod N_2\}$  for random  $v_j \in Z_{N_j}^*$ .  
 The prover sends  $V_1, V_2$ .
2. The verifier chooses a challenge bit  $d$  and sends it.
3. The prover responds by sending:  
 $\{m_1 = v_1 x_1^d \bmod N_1, m_2 = v_2 x_2^d \bmod N_2\}$
4. The verifier checks that there exists a bit  $a$  such that both:  
 $m_1^2 = (-1)^a \cdot V_1 \cdot X_1^d \bmod N_1$  and  $m_2^2 = (-1)^a \cdot V_2 \cdot X_2^d \bmod N_2$

The verifier accepts only if the checks succeed in all iterations.

**Fig. 4.** Proof of knowledge of “twin” GM-encryption

The above proof system is complete and sound; furthermore, it is easy to show that it is an honest-verifier zero-knowledge proof of knowledge (in fact, it remains honest-verifier zero-knowledge when the  $k$  iterations are run in parallel). To turn this to a non-interactive proof of knowledge in the random oracle model, we can use the standard Fiat-Shamir technique.

# Non-interactive Distributed-Verifier Proofs and Proving Relations among Commitments

Masayuki Abe<sup>1</sup>, Ronald Cramer<sup>2</sup>, and Serge Fehr<sup>2</sup>

<sup>1</sup> NTT Laboratories, Japan,  
abe@isl.ntt.co.jp

<sup>2</sup> BRICS\*, Aarhus University, Denmark,  
{cramer, fehr}@brics.dk

**Abstract.** A *commitment multiplication proof*, CMP for short, allows a player who is committed to secrets  $s$ ,  $s'$  and  $s'' = s \cdot s'$ , to prove, without revealing  $s$ ,  $s'$  or  $s''$ , that indeed  $s'' = ss'$ . CMP is an important building block for secure general multi-party computation as well as threshold cryptography.

In the standard cryptographic model, a CMP is typically done interactively using zero-knowledge protocols. In the random oracle model it can be done non-interactively by removing interaction using the Fiat-Shamir heuristic. An alternative non-interactive solution in the distributed setting, where at most a certain fraction of the verifiers are malicious, was presented in [1] for Pedersen's discrete log based commitment scheme. This CMP essentially consists of a few invocations of Pedersen's verifiable secret sharing scheme (VSS) and is secure in the standard model.

In the first part of this paper, we improve that CMP by arguing that a building block used in its construction in fact already constitutes a CMP. This not only leads to a simplified exposition, but also saves on the required number of invocations of Pedersen's VSS. Next we show how to construct non-interactive proofs of partial knowledge [8] in this distributed setting. This allows for instance to prove non-interactively the knowledge of  $\ell$  out of  $m$  given secrets, without revealing which ones. We also show how to construct efficient non-interactive zero-knowledge proofs for circuit satisfiability in the distributed setting.

In the second part, we investigate generalizations to other homomorphic commitment schemes, and show that on the negative side, Pedersen's VSS cannot be generalized to arbitrary (black-box) homomorphic commitment schemes, while on the positive side, commitment schemes based on  $q$ -one-way-group-homomorphism [7], which cover wide range of currently used schemes, suffice.

## 1 Introduction

Commitment schemes play an important role as a primitive in cryptographic protocols. Applications are found for instance in the construction of zero-knowledge

---

\* Basic Research in Computer Science ([www.brics.dk](http://www.brics.dk)), funded by the Danish National Research Foundation.

proofs and arguments, secure multi-party computation and threshold cryptography. Using a commitment scheme, a player can commit to a secret value  $s$  by publishing a *commitment*  $C$ , in such a way that the commitment  $C$  reveals nothing about the secret  $s$ , i.e., the scheme is *hiding*. The player can later *open*  $C$  to reveal  $s$  in a way verifiable by everyone else, i.e., it is *binding* in the sense that the player can't open  $C$  to any other value than  $s$ .

Many protocols using commitments require a player at some point to prove certain relations among a set of committed values, without revealing these committed values in the process. Assuming that addition or multiplication of secret values is well-defined, a player committed to  $s$ ,  $s'$  and  $s''$  will typically be required to prove that  $s'' = s + s'$  or that  $s'' = ss'$ . If the commitment scheme is homomorphic, as is the case with many known commitment schemes, the additive relation is trivial to handle, even non-interactively. A *commitment multiplication proof* (CMP), i.e., a secure protocol to handle the multiplicative relation, is generally less trivial to design.

In the two-player setting, there exist efficient *interactive* zero-knowledge protocols for all known homomorphic schemes [7]. These protocols can be adapted in a natural way to a distributed setting with  $n$  players and where up to  $t$  of them are malicious, for instance by simply letting each of the players be engaged in a separate run of the two-player protocol with the prover.

In [1] it is shown how this approach can be substantially improved by taking advantage of the fact that sufficiently many players are guaranteed to be honest. Namely, it is shown how to handle CMP in this distributed setting *non-interactively* in the case of Pedersen's discrete logarithm based commitment scheme. This CMP essentially consists of a few Pedersen VSS's and is non-interactive (from the prover's point of view) in case everyone plays honestly, while the prover might have to answer accusations otherwise. We call this *non-interactive with accusing*. Moreover, it is totally non-interactive if  $t < n/3$ .

In the first part of this paper, we improve that CMP by arguing that a building block used in its construction in fact already constitutes a CMP. This not only leads to a simplified exposition, but also saves on the required number of invocations of Pedersen's VSS. Next we show a new technique to construct non-interactive proofs of partial knowledge in this distributed setting, thereby extending the results of [8] for the interactive two-player case. This allows for instance to prove non-interactively the knowledge of  $\ell$  out of  $m$  given secrets, without revealing which ones. As an application, it allows to make the proof of correctness of a ballot in the [10] voting scheme non-interactive without resorting to random oracles. We also show how to construct efficient non-interactive zero-knowledge proofs for circuit satisfiability in the distributed setting.

In the second part, we investigate generalizations to other homomorphic commitment schemes, and show that on the negative side, Pedersen's VSS cannot be generalized to arbitrary (black-box) homomorphic commitment schemes, while on the positive side, commitment schemes based on  $q$ -one-way-group-homomorphism [7], which cover wide range of currently used schemes, suffice. Finally, we show how this positive result leads to error-free non-interactive zero-

knowledge proofs of membership for non-trivial languages in this distributed setting.

We proceed by repeating the concepts of commitment schemes and (verifiable) secret sharing and by recalling the concrete schemes of Pedersen in the following Section 2. In Section 3 we define (zero-knowledge) distributed-verifier proofs and we show that Pedersen's VSS can be seen as such a proof, and in Section 4 we present the CMP protocol and the proof protocols for partial knowledge and for general circuit satisfiability. Finally, in Section 5 we investigate to what extent the above protocols can be generalized to other homomorphic commitment schemes.

## 2 Preliminaries

### 2.1 Pedersen's Commitment Scheme

A *commitment scheme* of the kind we consider over a finite domain  $\mathcal{S}$  is given by a function family

$$\text{com}_{pk} : \mathcal{S} \times \mathcal{R}_{pk} \rightarrow \mathcal{C}_{pk}$$

indexed by a *public key*  $pk$ , where  $\mathcal{R}_{pk}$  and  $\mathcal{C}_{pk}$  are finite sets. In a set-up phase, a concrete public key  $pk$  and thus function  $\text{com}_{pk}$  is fixed in a prescribed manner. By publishing a *commitment*  $C = \text{com}_{pk}(s, r)$  for a random  $r \in \mathcal{R}_{pk}$ , such a scheme allows a party, Alice, to *commit* herself to a secret  $s \in \mathcal{S}$ , such that the commitment  $C$  reveals nothing about the secret  $s$  (*hiding property*) while on the other hand Alice can *open*  $C$  to  $s$  by publishing  $(s, r)$  *but only to  $s$*  (*binding property*).

If  $\mathcal{S}$  is a field  $K$  (or, more generally, a ring), then such a commitment scheme is called *homomorphic*, if the following holds: For any commitments  $C$  and  $C'$  and any number  $\lambda \in K$ , one can compute commitments  $S$  and  $P$  such that being able to open  $C$  and  $C'$  to values  $s$  and  $s'$ , respectively, allows to open  $S$  to the sum  $s + s'$  and  $P$  to the product  $\lambda s$ .

A well known homomorphic commitment scheme is the Pedersen commitment scheme [52, 16], given by

$$\begin{aligned} \text{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q &\rightarrow G \\ (s, r) &\mapsto g^s h^r \end{aligned}$$

where  $q$  is a prime,  $G$  is a (multiplicative) group of order  $|G| = q$  in which computing discrete logarithms is (assumed to be) hard, e.g. a subgroup of  $\mathbb{F}_p^*$ , and  $g$  and  $h$  are randomly chosen generators of  $G$ . This scheme is unconditionally hiding and computationally binding, and it is homomorphic: If  $C = \text{com}_{g,h}(s, r)$  and  $C' = \text{com}_{g,h}(s', r')$  then  $C \cdot C' = \text{com}_{g,h}(s + s', r + r')$  and  $C^\lambda = \text{com}_{g,h}(\lambda s, \lambda r)$ .

### 2.2 Pedersen's Verifiable Secret Sharing Scheme

In a *secret sharing scheme* a *dealer* distributes a *secret*  $s$  to  $n$  players  $P_1, \dots, P_n$  (for simplicity we set  $P_i = i$ ) by privately sending to each player  $P_i$  a *share*  $s_i$

in such a way that, for a fixed *threshold*  $t$ , up to  $t$  players have no information about the secret  $s$  (*privacy*) while  $t + 1$  players (or more) are able to reconstruct it (*correctness*). While secret sharing only guarantees security against curious players that try to gather information they are not supposed to obtain but otherwise behave honestly, its stronger version *verifiable secret sharing* [11], VSS for short, is secure in the following sense against up to  $t$  dishonest players and a possibly dishonest dealer that behave in an arbitrary manner.

*Privacy:* In case of an honest dealer, the information the dishonest players gain during the distribution of the secret  $s$  gives no information about  $s$ .

*Correctness:* As soon as the distribution is completed, there exists a fixed value  $s'$  such that every honest player will output  $s'$  as a result of the reconstruction, and if the dealer is honest, then  $s' = s$ .

The Pedersen VSS scheme [16] is based on Shamir's secret sharing scheme [17] and Pedersen's commitment scheme.

### Protocol Share <sub>$g,h$</sub>

1. To share a secret  $s \in \mathbb{F}_q$ , the dealer chooses a random polynomial  $f_s(X) = a_0 + a_1X + \dots + a_tX^t \in \mathbb{F}_q[X]$  of degree at most  $t$  with constant coefficient  $a_0 = s$ , and he commits himself to this *sharing polynomial*  $f_s(X)$  by broadcasting commitments  $A_0, \dots, A_t$  of  $a_0, \dots, a_t$ , respectively. For every player  $P_i$ , the dealer computes the share

$$s_i = f_s(i) = s + a_1i + \dots + a_ti^t \in \mathbb{F}_q$$

and he opens the corresponding commitment

$$C_i = A_0 \cdot A_1^i \cdot \dots \cdot A_t^{i^t}$$

privately to  $P_i$ , using the homomorphic property of the commitment scheme.

2. If  $P_i$  does not accept the opening, then  $P_i$  broadcasts an accusation against the dealer.
3. To any accusation of a player  $P_i$ , the dealer responds by opening  $C_i$  publicly.
4. If he fails to do this correctly then the sharing is rejected, otherwise it is accepted.

After the execution of this protocol, assumed that it has been accepted, every player  $P_i$  is committed to his share  $s_i$  by the commitment  $C_i$ , and he holds the corresponding information to open it. Hence, the reconstruction works as follows.

### Protocol Reconstruct <sub>$g,h$</sub>

Every player  $P_i$  publicly opens  $C_i$  to  $s_i$ . The shares  $s_i$  that have been correctly opened are then taken to reconstruct the secret by interpolation.

The pair (Share, Reconstruct) is a VSS if (and only if)  $t < n/2$ . Privacy holds unconditionally while correctness holds under the assumption that computing discrete logarithms is hard.

The scheme can be made completely non-interactive from the dealer's point of view in case  $t < n/3$  by replacing the steps 3 and 4 by

- 3.' If the number of accusations is larger than  $t$ , then the sharing is rejected, otherwise it is accepted.

Namely, in this case, if the sharing is accepted then there are at least  $t+1$  honest players that have not accused the dealer in step 2 and hence have a consistent sharing that allows to reconstruct the secret (see also the proof of Proposition 1).

*Remark.* Consider an accepted execution of the sharing protocol. By correctness, a secret value  $s'$  is fixed that can later be reconstructed. Since the information used by the players in the reconstruction originated with the dealer, we can conclude that the dealer knows this secret. In fact, it is straight forward to show, as we do later on, that the dealer not only knows  $s'$  but he also knows how to open the commitment  $A_0$  used in the sharing protocol (to  $s'$ ).

### 3 Distributed Verifier Proofs

#### 3.1 Model and Definition

We consider a *prover*  $P$  who wants to prove to a set of  $n$  *verifiers*  $\mathcal{V} = \{V_1, \dots, V_n\}$ , that he knows some witness  $w$  without revealing it. We assume an adversary that can actively corrupt up to  $t$  of the  $n$  verifiers as well as the prover  $P$ , where we consider both cases  $t < n/2$  and  $t < n/3$ . Some of the protocols require the adversary to be computationally bounded, and we assume him to be *static*, meaning that he has to corrupt the parties *before* the protocol execution. We assume that secure pairwise channels as well as broadcast channels are either provided by cryptographic means (in case of a bounded adversary) or given as primitives, though, for simplicity, also in the former case we will treat them as being perfectly secure.

Consider now two sets  $\mathcal{W}$  and  $\mathcal{I}$  and an efficiently verifiable relation  $R \subseteq \mathcal{W} \times \mathcal{I}$ . Given some *public information*  $I \in \mathcal{I}$ , the prover wants to convince the verifiers that he knows a *witness*  $w \in \mathcal{W}$  with  $(w, I) \in R$ .

**Definition 1.** A distributed verifier proof (of knowledge) for relation  $R$  is a protocol among a prover  $P$  and  $n$  verifiers  $V_1, \dots, V_n$  (all polynomially bounded), with a common input  $I$ , a private input  $w$  by  $P$  and a public output *accept* or *reject*, such that the following security properties hold, even if up to  $t$  of the  $n$  verifiers as well as the prover might be corrupted by the adversary.

*Correctness:* If  $P$  is honest and  $(w, I) \in R$ , then the output will be *accept*.

*Soundness:* There exists a knowledge extractor that can efficiently compute from the joint view of the honest players a witness  $w'$  satisfying  $(w', I) \in R$ , assumed that the output of the protocol is *accept*.

*This soundness condition can come in three flavors: perfect, unconditional, or computational. Meaning that the condition holds with probability 1, with overwhelming probability, or under some computational assumption, respectively.*

*A distributed verifier proof is called non-interactive, if the structure of the protocol is as follows. The prover sends to every verifier one message, a personal partial proof, and then every verifier votes to either accept or reject the proof, depending on whether he accepts or rejects his partial proof, and the outcome of the protocol is accept if and only if not more than  $t$  verifiers vote for rejection.*

*It is called non-interactive with accusing, if it is non-interactive except that in case there are some rejections, the prover must broadcast the corresponding partial proofs, and the outcome of the protocol is accept if and only if none of these published proofs is rejected.*

*Finally, it is called zero-knowledge, if the adversary can simulate his view of the protocol.*

The above soundness condition highlights the power of the distributed verifier setting in two ways: 1) The prover is *not* given to the knowledge extractor as a *rewindable* black-box. Thus, no rewinding argument is needed to prove the soundness of a protocol. 2) In case of *perfect* soundness it asserts that there is no knowledge error. Hence, acceptance of the proof always implies the knowledge of a witness  $w'$ . Of course, one can relax the definition by allowing to rewind the prover so that it becomes seamless with the standard definition of proof of knowledge [4] with a single verifier.

Such a distributed verifier proof can also be seen as a proof of membership where the prover proves the *existence* of a witness  $w$  with  $(w, I) \in R$  and therefore that  $I$  belongs to the language  $L_R = \{I \mid \exists w : (w, I) \in R\}$ . A proof of membership for language  $L$  in this model can be defined similarly, with the corresponding correctness and soundness conditions as follows.

*Correctness: If  $P$  is honest and  $x \in L$ , then the output will be accept.*

*Soundness: If the output of the protocol is accept, then  $x \in L$ .*

Again, soundness can come in different flavours. It is, however, important to note that in a usual single verifier proof perfect soundness can be achieved only for trivial languages while this is not true for distributed verifier proofs. This will be addressed further in Section 5.3. Proofs of membership in a distributed setting have also been introduced in [3] under the name of *network zero-knowledge proofs*.

### 3.2 Pedersen's VSS as a Distributed Verifier Proof

Let  $\text{com}_{g,h} : \mathbb{F}_q \times \mathbb{F}_q \rightarrow G$ ,  $(s, r) \mapsto g^s h^r$  be the Pedersen commitment scheme. For a commitment  $C = \text{com}_{g,h}(s, r)$  let  $\text{Proof}_{g,h}(C)$  denote an execution of  $\text{Share}_{g,h}$  with secret  $s$ , except that in step II of the protocol,  $A_0 = C$  is taken as commitment of  $a_0 = s$ . Then,  $\text{Proof}_{g,h}(C)$  is a zero-knowledge proof that the dealer can open the commitment  $C$ . More formally, for relation

$$R_{g,h} = \{((s, r), C) \mid s, r \in \mathbb{F}_q, C = \text{com}_{g,h}(s, r)\} \subseteq (\mathbb{F}_q)^2 \times G$$

we have

**Proposition 1.** *Protocol  $\text{Proof}_{g,h}$  is a perfectly sound zero-knowledge distributed-verifier proof for relation  $R_{g,h}$ , non-interactive in case  $t < n/3$  and non-interactive with accusing in case  $t < n/2$ .*

We stress that we have soundness and zero-knowledge independent of the quality of the commitment scheme  $\text{com}_{g,h}$ . In fact, this holds even in case the discrete logarithm  $\log_g h$  is known and hence the binding property does not hold at all.

*Proof of Proposition 1.* Since, to any possible accusation, the honest prover only broadcasts correct information, the proof will be accepted. It remains to show soundness and zero-knowledge.

*Soundness:* Assume that the proof has been accepted, and let  $A$  be the set of honest players, respectively, in case of  $t < n/3$ , the set of honest players who have not accused the dealer. In any case,  $|A| \geq t + 1$  (and we assume without loss of generality that  $A = \{1, \dots, t + 1\}$ ) and every player  $P_i \in A$  can open his commitment  $C_i$  to, say,  $s'_i$ . Let  $\lambda_1, \dots, \lambda_{t+1}$  be the *reconstruction coefficients* for the players in  $A$ . That is,  $\sum_{i=1}^{t+1} \lambda_i s_i = s$  for correctly computed shares  $s_i$  of  $s$ , which means that  $\sum_{i=1}^{t+1} \lambda_i \sum_{k=0}^t a_k i^k = s = a_0$  and hence  $\sum_{i=1}^{t+1} (\lambda_i i^k) = \delta_{k0}$  (where  $\delta_{ij} = 1$  if  $i = j$  and 0 otherwise). Because of the homomorphic property of the commitment scheme, the players of  $A$  can open the commitment  $C' = \prod_{i=1}^{t+1} C_i^{\lambda_i}$  to  $s' = \sum_{i=1}^{t+1} \lambda_i s'_i$ . However, as

$$C' = \prod_{i=1}^{t+1} C_i^{\lambda_i} = \prod_{i=1}^{t+1} \left( \prod_{k=0}^t A_k^{i^k} \right)^{\lambda_i} = \prod_{k=0}^t A_k^{\sum_i (\lambda_i i^k)} = A_0 = C \quad (1)$$

it follows that they can open  $C$  (to  $s'$ ).

*Zero-Knowledge:* Let  $A$  be the set of corrupted players. We assume without loss of generality that  $A = \{1, \dots, t\}$ . We make use of the well known fact that from the secret  $s$  and the shares  $s_1, \dots, s_t$  of the players in  $A$ , all the random sharing coefficients  $a_1, \dots, a_t$  can be computed in a linear way. Hence, writing  $s_0 = s = a_0$ , for every  $k \in \{0, \dots, t\}$  there exist coefficients  $\mu_{k0}, \dots, \mu_{kt}$  such that  $a_k = \sum_{j=0}^t \mu_{kj} s_j$ , which means that  $s_i = \sum_{k=0}^t a_k i^k = \sum_{k=0}^t \sum_{j=0}^t \mu_{kj} s_j i^k$  and hence  $\sum_{k=0}^t \mu_{kj} i^k = \delta_{ij}$ .

Given the commitment  $C$  for  $s$ , the players in  $A$  can simulate their view of the protocol as follows. For every  $P_i \in A$  they choose  $s_i \in \mathbb{F}_q$  at random and compute a (random) commitment  $C_i$  for  $s_i$ , and for  $k = 0, \dots, t$  they compute  $A_k = \prod_{j=0}^t C_j^{\mu_{kj}}$ , where  $C_0 = C$ , such that  $A_0 = C$  and for every  $i \in A$

$$\prod_{k=0}^t A_k^{i^k} = \prod_{k=0}^t \left( \prod_{j=0}^t C_j^{\mu_{kj}} \right)^{i^k} = \prod_{j=0}^t C_j^{\sum_k (\mu_{kj} i^k)} = C_i \quad (2)$$

Finally, it is not hard to see that  $A_1, \dots, A_t$  are independently random commitments of independently random values.  $\square$



## 4 Our Technical Contributions

### 4.1 An Improved Commitment Multiplication Proof

Consider again Pedersen's commitment scheme  $\text{com}_{g,h}(s, r) = g^s h^r$ , and let  $C' \in G$  be an arbitrary commitment. Then the commitment scheme

$$\text{com}_{C',h}(s^*, r^*) = (C')^{s^*} h^{r^*} = C'^{s^*} \cdot \text{com}_{g,h}(0, r^*)$$

with basis  $C', h$  inherits the following properties.

**Lemma 1.**

1. *Being able to open (wrt.  $\text{com}_{g,h}$ )  $C'$  and  $C''$  to values  $s'$  and  $s''$ , respectively, allows to open  $C''$  wrt.  $\text{com}_{C',h}$  to a value  $s$  satisfying  $ss' = s''$ , and being able to open  $C''$  to 0 wrt.  $\text{com}_{g,h}$  allows to open  $C''$  to 0 wrt.  $\text{com}_{C',h}$ .*
2. *The scheme  $\text{com}_{C',h}$  is as hiding and binding as  $\text{com}_{g,h}$ , assumed that  $C'$  cannot be opened to 0 wrt.  $\text{com}_{g,h}$ .*

*Proof.* 1. Let  $s, s', s'', r', r''$  satisfy  $C' = \text{com}_{g,h}(s', r')$ ,  $C'' = \text{com}_{g,h}(s'', r'')$  and  $ss' = s''$ . Then, for  $r^* = r'' - sr'$  we have  $\text{com}_{g,h}(s'' - ss', r^*) = C'' \cdot C'^{-s}$  and hence  $\text{com}_{C',h}(s, r^*) = C'^s \cdot \text{com}_{g,h}(0, r^*) = C''$ . This also holds if  $s = 0$  and thus  $s'' = 0$ , in which case  $r^* = r''$ .

2. First, for  $r^* \in \mathbb{F}_q$  chosen at random,  $\text{com}_{C',h}(s^*, r^*)$  is clearly a random element of  $G$ , independent of  $s^*$ . Furthermore, knowing  $s^* \neq \tilde{s}^*$  and  $r^*$  and  $\tilde{r}^*$  such that  $\text{com}_{C',h}(s^*, r^*) = \text{com}_{C',h}(\tilde{s}^*, \tilde{r}^*)$ , i.e.  $C'^{s^*} \cdot \text{com}_{g,h}(0, r^*) = C'^{\tilde{s}^*} \cdot \text{com}_{g,h}(0, \tilde{r}^*)$ , allows to open the commitment  $C'$  to zero, namely  $C' = \text{com}_{g,h}(0, (r^* - \tilde{r}^*)/(\tilde{s}^* - s^*))$ .  $\square$

This gives rise to the following CMP, which allows the prover to prove that he can open commitments  $C, C'$  and  $C''$  to values  $s, s'$  and  $s'' = ss'$ , respectively. Note that the [4](#) steps can be executed in parallel.

**Protocol Mult Proof <sub>$g,h$</sub> ( $C, C'; C''$ )**

1. The prover executes **Proof <sub>$g,h$</sub> ( $C$ )**.
2. The prover executes **Proof <sub>$C',h$</sub> ( $C''$ )** using the *same* sharing polynomial  $f_s(X)$  as in the above step (but new independent commitments wrt.  $\text{com}_{C',h}$ ).
3. Every player verifies whether his shares from step 1. and 2. coincide and accuses the dealer if it does not hold. In case  $t < n/2$  (but not  $t < n/3$ ) the dealer responds by opening the two corresponding commitments in public.
4. The prover executes **Proof <sub>$g,h$</sub> ( $C''$ )**.

This protocol also appeared in [\[1\]](#). However, the security proof given there did *not* cover the case where the prover can open  $C'$  to  $s' = 0$ , and therefore the protocol was extended to “also deal with the case  $s' = 0$ ” by essentially adding another Pedersen VSS sharing. Our analysis shows that this is superfluous, and that the protocol as it stands is secure also in case  $s' = 0$ . Furthermore, we show that the case  $s = 0$  is somewhat special. Namely, we show that if the prover can

open the commitment  $C$  to  $s = 0$ , then he can execute the protocol even *without being able to open  $C'$* , as long as he can open  $C''$  to  $s'' = 0$ . This of course also guarantees that  $s'' = ss'$  (no matter what  $s'$  is), but, as we will see in the next section, it also opens the door for new constructions in this setting like proofs of partial knowledge.

**Theorem 1.** *The above protocol  $\text{MultProof}_{g,h}(C, C'; C'')$  is a perfectly sound zero-knowledge distributed verifier proof, non-interactive in case  $t < n/3$  and non-interactive with accusing in case  $t < n/2$ , that the prover can open  $C$ ,  $C'$  and  $C''$  as values  $s$ ,  $s'$  and  $s'' = ss'$ , or that he can open both  $C$  and  $C''$  as 0.*

*Proof. Correctness:* Follows from point 1. of Lemma 1 and the correctness of the protocol  $\text{Proof}_{g,h}$ .

*Soundness:* According to Proposition 1, from the information received during Step 1., the honest players can compute  $s$  and  $r$  with  $C = \text{com}_{g,h}(s, r)$ . Also, from the information received during Step 2., the honest players can compute the same  $s$  and some  $r^*$  with  $C'' = \text{com}_{C',h}(s, r^*) = C'^s \cdot \text{com}_{g,h}(0, r^*)$ . Finally, from the information received during Step 3., the honest players can compute  $s''$  and  $r''$  with  $C'' = \text{com}_{g,h}(s'', r'')$ . It now follows that either  $s = 0$  and hence  $C'' = \text{com}_{g,h}(0, r^*)$ , which means that the honest players can open  $C''$  to zero, or that  $C' = C''^{1/s} \cdot \text{com}_{g,h}(0, r^*)^{-1/s} = \text{com}_{g,h}(s'', r'')^{1/s} \cdot \text{com}_{g,h}(0, r^*)^{-1/s} = \text{com}_{g,h}(s''/s, (r'' - r^*)/s)$ , which means that the honest players can open  $C'$  to  $s' = s''/s$ .

*Zero-Knowledge:* The adversary can simulate his view of the protocol by simulating independently the protocols  $\text{Proof}_{g,h}(C)$ ,  $\text{Proof}_{C',h}(C'')$  and  $\text{Proof}_{g,h}(C'')$ , as described in the proof of Proposition 1 *except* that he chooses the same shares for the simulation of  $\text{Proof}_{g,h}(C)$  and of  $\text{Proof}_{C',h}(C'')$ .  $\square$

## 4.2 Proofs of Partial Knowledge

In [8], an efficient solution was presented to construct proofs of *partial knowledge* in the two-players setting. Such a proof of partial knowledge allows for instance to prove the knowledge of (at least)  $\ell$  out of  $m$  given secrets without revealing which  $\ell$  secrets. We will now present corresponding non-interactive protocols in the distributed-verifier setting. While the proof protocols of [8] rely on concepts like the dual access structure and the simulation of protocols, our distributed verifier proof protocols are based on the fact that the CMP protocol  $\text{MultProof}_{g,h}(C, C'; C'')$  can be executed by the prover even if he does not know  $s'$  (as long as  $s = s'' = 0$ ).

Let first  $C_0$  and  $C_1$  be two public Pedersen commitments and let the prover be able to open  $C_w$  to say  $s_w$ , where either  $w = 0$  or  $w = 1$ .

### Protocol OR-Proof $_{g,h}(C_0, C_1)$

The prover sets  $b_w = 1$  and  $b_{1-w} = 0$  as well as  $d_w = s_w$  and  $d_{1-w} = 0$ , and he commits to  $b_0, b_1, d_0$  and  $d_1$  by  $B_0, B_1, D_0$  and  $D_1$ , respectively. Then, he opens  $B = B_0 \cdot B_1$  as  $b_0 + b_1 = 1$  and executes  $\text{MultProof}_{g,h}(B_0, C_0; D_0)$  and  $\text{MultProof}_{g,h}(B_1, C_1; D_1)$ .

According to Theorem 1, the prover can execute  $\text{Mult Proof}_{g,h}(B_{1-w}, C_{1-w}; D_{1-w})$  even without being able to open  $C_{1-w}$  as long as he can open  $B_{1-w}$  and  $D_{1-w}$  to zero. On the other hand, if he cannot open  $B_w$  to zero, which must be the case for at least one of  $B_0$  and  $B_1$  as he can open  $B = B_0 \cdot B_1$  to 1,  $\text{Mult Proof}_{g,h}(B_w, C_w; D_w)$  proves that he can open  $C_w$ .

This can easily be generalized to  $\ell$ -out-of- $m$  proofs, which, given  $m$  commitments  $C_1, \dots, C_m$ , allows to prove the knowledge of at least  $\ell$  hidden secrets, without giving away which ones.

### Protocol $(\binom{\ell}{m})\text{-Proof}_{g,h}(C_1, \dots, C_m)$

For  $i = 1, \dots, m$ , the prover sets  $b_i = 1$  and  $d_i = s_i$  if he can open  $C_i$  (to  $s_i$ ) and  $b_i = d_i = 0$  otherwise, and he commits to  $b_i$  and  $d_i$  by  $B_i$  and  $D_i$ , respectively. He proves that indeed  $b_i \in \{0, 1\}$ , i.e.  $b_i(1 - b_i) = 0$ , by executing  $\text{Mult Proof}_{g,h}(B_i, E/B_i; O)$ , where  $E = \text{com}_{g,h}(1, 0) = g$  and  $O = \text{com}_{g,h}(0, 0) = 1$  are default commitments for 1 and zero, respectively, he opens  $B_1 \dots B_m$  as  $\ell$  and executes  $\text{Mult Proof}_{g,h}(B_i, C_i; D_i)$  for  $i = 1, \dots, m$ .

The following is a somewhat more efficient solution where no proof of something like  $b_i \in \{0, 1\}$  is needed. Consider Shamir's  $\ell$ -out-of- $m$  secret sharing scheme. As we have already used in the proof of Proposition 1 for  $A \subseteq \{1, \dots, m\}$  with  $|A| \geq \ell$ , there exist reconstruction coefficients  $\lambda_{A,i}$ ,  $i \in A$ , such that  $\sum_{i \in A} (\lambda_{A,i} i^k) = \delta_{k0}$ . Based on this fact, we have the following enhanced protocol that allows the prover to prove that he can open the commitments  $C_i$  with  $i \in A$  for a subset  $A \subseteq \{1, \dots, m\}$  of size at least  $\ell$ .

### Protocol $(\binom{\ell}{m})\text{-Proof}'_{g,h}(C_1, \dots, C_m)$

The prover chooses reconstruction coefficients  $\lambda_{A,i}$ ,  $i \in A$ . For  $i = 1, \dots, m$ , he puts  $b_i = \lambda_{A,i}$  and  $d_i = b_i s_i$  if  $i \in A$  and  $b_i = d_i = 0$  otherwise, and he generates commitments  $B_1, \dots, B_m$  and  $D_1, \dots, D_m$  for  $b_1, \dots, b_m$  and  $d_1, \dots, d_m$ , respectively. For  $k = 0, \dots, \ell$ , he opens the commitment  $\prod_{i=1}^m B_i^{i^k}$  as  $\delta_{k0}$ , and he executes  $\text{Mult Proof}_{g,h}(B_i, C_i; D_i)$  for  $i = 1, \dots, m$ .

Soundness of the above protocol relies on the binding property of the Pedersen commitment scheme (hence it allows small error probability).

It is not hard to see that this protocol can be generalized to any linear secret sharing scheme, not necessarily a threshold scheme. Hence, given an arbitrary linear secret sharing scheme over  $\mathbb{F}_q$  for  $m$  players with an access structure  $\Gamma$ , we have the following

**Theorem 2.** *Under the DL-assumption, there exists a computationally sound zero-knowledge distributed-verifier proof, non-interactive in case  $t < n/3$  and non-interactive with accusing in case  $t < n/2$ , that the prover can open a subset  $C_{i_1}, \dots, C_{i_\ell}$  of the commitments  $C_1, \dots, C_m$  corresponding to a qualified set  $A = \{i_1, \dots, i_\ell\} \in \Gamma$ .*

### 4.3 General Circuit Evaluation Proofs

Let  $\mathcal{C}$  be a binary circuit consisting of NAND gates.

**Theorem 3.** *Under the DL-assumption, there exists a computationally sound zero-knowledge distributed-verifier proof, non-interactive in case  $t < n/3$  and non-interactive with accusing in case  $t < n/2$ , that the prover knows a satisfying input to the circuit  $\mathcal{C}$ .*

*Proof Sketch:* Let  $b = (b_1, \dots, b_m)$  be a satisfying input for the circuit  $\mathcal{C}$ . To prove knowledge of  $b$ , the prover generates a commitment  $B_i$  for every input bit  $b_i$  and proves that  $b_i \in \{0, 1\}$  by executing  $\text{MultProof}_{g,h}(B_i, E/B_i; O)$ . Inductively, for every NAND gate with input bits  $b_l$  and  $b_r$  to which he has already computed corresponding commitments  $B_l$  and  $B_r$ , respectively, the prover computes a commitment  $B_{out}$  for the output bit  $b_{out} = b_l \text{ NAND } b_r$  and proves its correctness by executing  $\text{MultProof}_{g,h}(B_l, B_r; E/B_{out})$ . Finally, he opens the commitment  $B$  of the result bit  $b = \mathcal{C}(b_1, \dots, b_m)$  as 1.  $\square$

Another way to achieve this result is by combining the techniques from [6] based on proofs of partial knowledge with the protocols from the above section.

Clearly, if the circuit  $\mathcal{C}$  is an arithmetic circuit over the field  $\mathbb{F}_q$ , then there exists an even simpler proof protocol.

## 5 Arbitrary Homomorphic Commitments

In this section, we investigate to what extent the Pedersen's VSS scheme and the above results can be generalized with regard to other homomorphic commitment schemes. Clearly, by the description in Section 2.2, the Pedersen's VSS scheme, consisting of the protocols **Share** and **Reconstruct**, can be executed with an arbitrary homomorphic commitment scheme replacing the Pedersen scheme. However, it is not so clear whether this results in a *secure* VSS scheme. And indeed, we will show that the security cannot be proven for an arbitrary (black-box) homomorphic commitment scheme. This does not necessarily imply that there exists a secure commitment scheme under which the Pedersen-like VSS is insecure; however, it means that in order to result in a secure Pedersen-like VSS, a homomorphic commitment scheme must inherit some additional properties. On the other hand, to relax the impact of this negative result, we present sufficient conditions for a homomorphic commitment scheme that guarantee the security of the corresponding Pedersen-like VSS and the resulting distributed-verifier proofs. We then show that these conditions are satisfied by so called  $q$ -one-way-group-homomorphism based schemes [7], which cover all currently known homomorphic commitment schemes with finite domain. Finally, we show how this positive result leads to error-free non-interactive zero-knowledge proofs of membership.

### 5.1 The Impossibility Result

Recall that a commitment scheme over a field  $K$  is called homomorphic if, given two commitments  $C$  and  $C'$  and a field element  $\lambda \in K$ , one can compute commitments  $S$  and  $P$  such that being able to open  $C$  and  $C'$  to values  $s$  and  $s'$ , respectively, allows to open  $S$  to  $s + s'$  and  $P$  to  $\lambda s$ . We will denote these mappings  $(C, C') \mapsto S$  and  $(\lambda, C) \mapsto P$  by “ $\star$ ” and “ $\circ$ ”, respectively, i.e. we write  $S = C \star C'$  and  $P = \lambda \circ C$ . The following theorem states that the Pedersen VSS scheme described in Section 2.2 cannot be generalised to a homomorphic commitment scheme  $\text{com}$ , that is given as a *black-box* and where only the security requirements and the homomorphic property are guaranteed. The idea is that with respect to some unconditionally-hiding homomorphic commitment scheme, the dealer might be able to come up with commitments  $A_0 = C, A_1, \dots, A_t$  for the secret and the sharing coefficients, computed in some way such that he is not able to open (all of) them, but nevertheless he can open the corresponding commitments  $C_1, \dots, C_n$  to a set  $s_1, \dots, s_n$  of inconsistent shares. This is for instance the case if the dealer can compute a commitment  $A_1$  such that he can open  $2 \circ A_1, \dots, (n-1) \circ A_1$  to  $2, \dots, n-1$ , respectively, such that it looks as if  $A_1$  “contains” 1, and  $n \circ A_1$  to, let’s say,  $n+1$ . Indeed, by choosing  $A_1$  this way and  $A_0 = C$  and  $A_2, \dots, A_t$  as required by the Share protocol, the dealer could open the corresponding commitments  $C_2, \dots, C_n$ , computed as  $C_i = C \star (i \circ A_1) \star \dots \star (i^t \circ A_t)$ , to a set of inconsistent shares (though he cannot open  $C_1$ ). Since we do not require the dealer to be able to open  $A_1$ , and the homomorphic property does not require anything like  $\lambda^{-1} \circ (\lambda \circ C) = C$  (as can be observed for existing schemes, see Section 5.2), the existence of such a commitment  $A_1$  does not *a priori* contradict the security of the commitment scheme, if it is unconditionally hiding and hence a statement like “ $A_1$  contains 1” does not make sense. We will now show that also *a posteriori*, this does not contradict the security (or the homomorphic property) of the commitment scheme by presenting an oracle with respect to which there exists a secure homomorphic commitment scheme, however the corresponding Pedersen-like VSS is insecure.

**Theorem 4.** *Let  $K$  be a field of size  $2^k$ , where  $k$  is a security parameter. There exists an oracle  $\mathcal{O}$  relative to which there exists a secure homomorphic commitment scheme  $\text{com}_{\mathcal{O}} : K \times K \rightarrow K$  such that the resulting Pedersen-like VSS, consisting of  $\text{Share}_{\mathcal{O}}$  and  $\text{Reconstruct}_{\mathcal{O}}$ , is insecure.*

The oracle  $\mathcal{O}$  in mind has history tapes  $\mathcal{H}$ ,  $\mathcal{M}$  and  $\mathcal{A}$ , which are all empty at the beginning, and one can make *commit*-, *multiply*-, *add*- and *cheat*-queries, to which  $\mathcal{O}$  answers as follows:

*commit-query:* input  $s, r \in K$ , output  $C = \text{com}_{\mathcal{O}}(s, r) \in K$

If there exists  $C \in K$  such that  $(s, r; C) \in \mathcal{H}$ , then  $\mathcal{O}$  returns  $C$ . Else,  $\mathcal{O}$  chooses a random  $C \in K$ , writes  $(s, r; C)$  to the history tape  $\mathcal{H}$  and returns  $C$ .

*multiply-query:* input  $\lambda, C \in K$ , output  $C' = \text{multiply}_{\mathcal{O}}(\lambda, C) \in K$

If there exists  $C' \in K$  such that  $(\lambda, C; C') \in \mathcal{M}$ , then  $\mathcal{O}$  returns  $C'$ . Else, if there exists  $s, r \in K$  such that  $(s, r; C) \in \mathcal{H}$ , then  $\mathcal{O}$  computes  $C' =$

$\text{com}_{\mathcal{O}}(\lambda s, \lambda r)$ , while otherwise it chooses  $C' \in K$  at random, and it writes  $(\lambda, C; C')$  to the history tape  $\mathcal{M}$  and returns  $C'$ .

*add-query:* input  $C, C' \in K$ , output  $C'' = \text{add}_{\mathcal{O}}(C, C') \in K$

If there exists  $C'' \in K$  such that  $(C, C'; C'') \in \mathcal{A}$ , then  $\mathcal{O}$  returns  $C''$ . Else, if there exist  $s, r, s', r' \in K$  such that  $(s, r; C), (s', r'; C) \in \mathcal{H}$ , then  $\mathcal{O}$  computes  $C'' = \text{com}_{\mathcal{O}}(s + s', r + r')$ , while otherwise it chooses  $C' \in K$  at random, and it writes  $(C, C'; C'')$  to the history tape  $\mathcal{A}$  and returns  $C''$ .

*cheat-query:* input  $n \in \mathbb{N}$ , output  $(r^{(2)}, \dots, r^{(n)}; C, C^{(2)}, \dots, C^{(n)}) \in K^{n-1} \times K^n$   
 $\mathcal{O}$  chooses random  $r^{(2)}, \dots, r^{(n)} \in K$  and  $C, C^{(2)}, \dots, C^{(n)} \in K$ . For  $i = 2$  to  $n$ , he writes  $(i, C; C^{(i)})$  to the history tape  $\mathcal{M}$ . For  $i = 2$  to  $n - 1$ , he writes  $(i, r^{(i)}; C^{(i)})$  to the history tape  $\mathcal{H}$ , and he writes  $(n + 1, r^{(n)}; C^{(n)})$  to the history tape  $\mathcal{H}$ . Finally, he returns  $r^{(2)}, \dots, r^{(n)}$  and  $C, C^{(2)}, \dots, C^{(n)}$ .

This oracle gives indeed rise to a homomorphic commitment scheme  $\text{com}_{\mathcal{O}} : K \times K \rightarrow K$ . Namely, as indicated by the notation, for  $s, r \in K$ , the commitment  $\text{com}_{\mathcal{O}}(s, r)$  is the answer of the oracle  $\mathcal{O}$  to a commit-query with input  $s$  and  $r$ , and the multiply- and add-queries provide the homomorphic property. E.g. being able to open  $C$  to  $s$ , i.e. knowing  $r$  such that  $(s, r; C) \in \mathcal{H}$ , allows to open  $\lambda \circ C = \text{multiply}_{\mathcal{O}}(\lambda, C)$ , the answer  $C'$  to a multiply-query with input  $\lambda$  and  $C$ , to the value  $\lambda s$ , since after the query  $(\lambda s, \lambda r; C') \in \mathcal{H}$  and hence  $\text{com}_{\mathcal{O}}(\lambda s, \lambda r) = C'$ . Furthermore, the cheat-query allows the dealer (together with a corrupted first player  $P_1$ ) to misbehave as described in the beginning of this section to distribute an inconsistent sharing among the remaining players  $P_2, \dots, P_n$ . It remains to show the security of  $\text{com}_{\mathcal{O}}$ . The commitment  $C$  of a secret  $s$ , generated with whatever query, is a random number in  $K$ , independent of anything else, and hence the scheme is hiding. Because of the same reason,  $C \neq C'$  for every pair  $(s, r, C), (s', r', C')$  of entries of  $\mathcal{H}$ , except with small probability, and hence the scheme is binding.

It is not hard to see from the above construction that with respect to this homomorphic commitment scheme  $\text{com}_{\mathcal{O}}$ , Proposition 1 and similarly Theorem 2 to 3 do not hold.

## 5.2 Generalization to $q$ -OWGH-Based Commitments

Inspecting for instance the proof of Proposition 1, which is essentially identical to a security proof of Pedersen's VSS scheme, one immediately sees that we made extensive use of the fact that for Pedersen's commitment scheme the operation “ $\star$ ” is a group operation “ $\cdot$ ”, and that “ $\circ$ ”, given by exponentiation, fulfils

$$(C \cdot C')^{\lambda} = C^{\lambda} \cdot C'^{\lambda}, \quad C^{\lambda+\lambda'} = C^{\lambda} \cdot C^{\lambda'} \quad \text{and} \quad C^{\lambda\lambda'} = (C^{\lambda})^{\lambda'}$$

which may not hold for other homomorphic schemes. In fact, with respect to the schemes listed in the appendix, this holds *only* for Pedersen's. For instance, if  $C$  is a commitment with respect to the QR-based commitment scheme  $\text{com}_t(s, r) = t^s r^2$  over  $\mathbb{F}_2 = \{\bar{0}, \bar{1}\}$  ( $\bar{x}$  denotes the residue class of  $x$  modulo  $q$  hereafter), then in general  $C^{\bar{1}} \cdot C^{\bar{1}} = C \cdot C = C^2 \neq 1 = C^{\bar{0}} = C^{\bar{1}+\bar{1}}$ . On the other hand,

it is not hard to see that these were the *sole* conditions needed (besides the homomorphic property), not only for the proof of Proposition 1, but also for all results from the Sections 4. Hence, the above properties give a sufficient condition for a homomorphic commitment scheme in order to generalize the security of Pedersen's VSS scheme as well as our results to this commitment scheme. And, as a matter of fact, even some weaker condition suffices (which, by the way, are fulfilled by the above QR-based scheme, as will be shown):

*It should be feasible to open the commitments*

$$(C \cdot C')^\lambda / (C^\lambda \cdot C'^\lambda), \quad C^{\lambda+\lambda'} / (C^\lambda \cdot C'^{\lambda'}) \quad \text{and} \quad C^{\lambda\lambda'} / (C^\lambda)^{\lambda'} \quad (3)$$

to zero for any commitments  $C, C'$  and numbers  $\lambda, \lambda'$ , knowing only  $C, C'$ ,  $\lambda$  and  $\lambda'$ .

Indeed, consider for instance (1) in the proof of Proposition 1. Even though it might be that  $C' \neq C$ , it is guaranteed by these properties that the commitment  $C/C'$  can be opened to zero knowing only  $C$  and  $C'$ , and hence being able to open  $C'$  (to  $s'$ ) also allows to open  $C = (C/C') \cdot C'$  (to  $s'$ ). This kind of reasoning allows to generalize all the previous proofs, and hence we have

**Proposition 2.** *The security of Pedersen's VSS scheme as well as Proposition 1 and Theorem 1 to 3 hold for every homomorphic commitment scheme satisfying the above condition (3).*

We will now show that all  $q$ -one-way-group-homomorphism based commitment schemes, which contain all so far known homomorphic schemes with finite domain, fulfil this condition (3). We start by recalling the concept of  $q$ -one-way-group-homomorphism. Let  $q$  be a prime number. Loosely speaking, a  $q$ -one-way-group-homomorphism,  $q$ -OWGH for short, is a homomorphism  $f : H \rightarrow G$  among two finite Abelian groups  $H$  and  $G$ , such that  $f$  is one-way, but, for a randomly chosen  $y \in G$ , it is feasible to compute  $v \in H$  with  $f(v) = y^q$ . For formal definitions we refer to [7], where this concept was introduced.

Such a  $q$ -OWGH induces in a generic way a computationally binding commitment scheme over the field  $\mathbb{F}_q$ . Namely the scheme

$$\text{com}_{g,f} : \mathbb{F}_q \times H \rightarrow G, \quad (s, r) \mapsto g^s f(r)$$

where  $g$  is randomly chosen from  $\text{im}(f) \subseteq G$  and  $g^s$  is defined as  $g^\varsigma$  with  $\varsigma \in \{0, \dots, q-1\}$  such that  $\bar{\varsigma} = s$ . Note, it is *not* required that  $G$  has order  $q$ .

If a  $q$ -OWGH  $f : H \rightarrow G$  is *unconditionally binding* [7], meaning that there exists  $t \in G$  such that  $t$  has order  $q$  modulo  $\text{im}(f)$  and  $t^i f(r)$  and  $t^j f(s)$  are computationally indistinguishable for all  $i$  and  $j$  and for randomly (and independently) chosen  $r$  and  $s$ , then  $f$  also induces a computationally hiding commitment scheme over  $\mathbb{F}_q$ . Namely,

$$\text{com}_{t,f} : \mathbb{F}_q \times H \rightarrow G, \quad (s, r) \mapsto t^s f(r)$$

for such a particular  $t \in G$ .

For the security proof of these commitments, we refer to [7].

An important property of these commitment schemes is that they are homomorphic. Indeed, if  $C = \text{com}(s, r) = g^s f(r)$  and  $C' = \text{com}(s', r) = g^{s'} f(r')$  and  $\lambda \in \mathbb{F}_q$ , we have, writing  $s = \bar{\varsigma}$ ,  $s' = \bar{\varsigma}'$ ,  $\lambda s + s' = \bar{\varsigma}''$  and  $\lambda = \bar{\ell}$  with  $\varsigma, \varsigma', \varsigma'', \ell \in \{0, \dots, q-1\}$  as well as  $\ell\varsigma + \varsigma' = kq + \varsigma'' \in \mathbb{Z}$ ,

$$\begin{aligned} C^\lambda C' &= (g^{\bar{\varsigma}} f(r))^\ell g^{\bar{\varsigma}'} f(r') = g^{\ell\varsigma + \varsigma'} f(\ell r + r') \\ &= g^{kq + \varsigma''} f(\ell r + r') = g^{\lambda s + s'} f(kv + \ell r + r') \end{aligned}$$

where  $v \in H$  is computed such that  $f(v) = g^q$ .

**Lemma 2.** *For any  $q$ -OWGH based commitment scheme, any commitments  $C$  and  $C'$  and numbers  $\lambda, \lambda' \in \mathbb{F}_q$ , the commitments*

$$(C \cdot C')^\lambda / (C^\lambda \cdot C'^\lambda), \quad C^{\lambda + \lambda'} / (C^\lambda \cdot C'^{\lambda'}) \quad \text{and} \quad C^{\lambda \lambda'} / (C^\lambda)^{\lambda'}$$

can be opened to zero knowing only  $C$ ,  $C'$ ,  $\lambda$  and  $\lambda'$ .

*Proof.* Clearly,  $(C \cdot C')^\lambda = (C^\lambda \cdot C'^\lambda)$  and thus  $(C \cdot C')^\lambda / (C^\lambda \cdot C'^\lambda) = 1 = \text{com}_{g,f}(0, 0)$ . Furthermore, if  $\lambda = \bar{\ell}$ ,  $\lambda' = \bar{\ell}'$  and  $\lambda\lambda' = \bar{\ell}''$  with  $\ell, \ell', \ell'' \in \{0, \dots, q-1\}$  and  $\ell\ell' = kq + \ell''$ , we get

$$C^{\lambda \lambda'} / (C^\lambda)^{\lambda'} = C^{\ell' - \ell\ell'} = C^{-kq} = f(-kv) = \text{com}_{g,f}(0, -kv)$$

where  $v \in H$  is computed such that  $f(v) = C^q$ . And of course, the same argument can be applied to  $C^{\lambda + \lambda'} / C^\lambda C'^{\lambda'}$ .  $\square$

It now follows from Proposition 2

**Theorem 5.** *The security of Pedersen's VSS scheme as well as Proposition 1 and Theorems 1 to 3 hold for every  $q$ -OWGH based commitment scheme.*

Note that Shamir's secret sharing scheme does not work over  $\mathbb{F}_q$  if  $q \leq n$ . Hence, in this case, Pedersen's VSS and the resulting proof protocols have to be based on a different linear secret sharing scheme. However, it is straight forward to verify that replacing Shamir's secret sharing scheme in Pedersen's VSS and the resulting proof protocols by an arbitrary linear secret sharing scheme [14] does not affect any of the results. This also allows to generalize the results to arbitrary (not necessarily threshold) adversary structures [13].

### 5.3 On Proofs of Membership

In this last section, we show that in the distributed-verifier setting there exist error-free non-interactive zero-knowledge proofs of membership for non-trivial languages, which is well known not to exist in the usual single-verifier setting.

Recall the protocol Proof from Section 3.2 but now based on an arbitrary  $q$ -OWGH based commitment scheme  $\text{com} : \mathbb{F}_q \times H \rightarrow G$ . It allows the dealer to prove that he can open a given commitment  $C$  to *some* value. Assume now that



he wants to prove that he can open  $C$  to a concrete given value  $s$ , e.g.  $s = 0$ . This can be done simply by executing the protocol **Proof** as given, except that the dealer uses the default sharing polynomial  $f_s(X) = s$  (instead of a random one), such that every share coincides with  $s$  (and if this is not the case for some player then he accuses the dealer). We denote this modified protocol by **Proof'**. It can be shown similarly to the proof of Proposition 1 that this indeed proves that the dealer can open  $C$  to  $s$ , or, in terms of proofs of membership, that  $C$  is a commitment of  $s$ :

**Proposition 3.** *Protocol **Proof'** is a perfectly sound zero-knowledge distributed-verifier proof that  $C$  is in  $\{\text{com}(s, r) \mid r \in H\} \subseteq G$ , non-interactive in case  $t < n/3$  and non-interactive with accusing in case  $t < n/2$ .*

Using unconditionally binding commitment schemes like the QR- or the DCR-based ones described in the appendix, this results in error-free non-interactive proofs for non-trivial subgroup membership problems: The former allows to prove that a given number is a quadratic residue modulo an RSA modulus  $n$ , and the latter that a given number is an  $n$ -th power modulo  $n^2$ , simply by proving that the number is a commitment of  $s = 0$ .

In fact, one can construct proves for arbitrary subgroup membership problems (even if they do not result from homomorphic commitments), i.e., proves that allow to prove that a group element  $C \in G$  belongs to a subgroup  $G' \subset G$ , as long as for every  $C \in G'$  there exists a corresponding witness  $w$  in a group  $H$  such that the mapping  $\varphi : H \rightarrow G'$ ,  $w \mapsto C$  is a group homomorphism. Namely, by executing **Proof** using  $\text{com} = \varphi : H \times \emptyset \rightarrow G'$  as “commitment”: The dealer chooses random witnesses  $a_1, \dots, a_t \in H$ , publishes the corresponding subgroup elements  $A_k = \varphi(a_k) \in G'$ ,  $k = 1, \dots, t$ , and sends the witness for  $C_i = C \cdot A_1^{i_1} \cdot \dots \cdot A_t^{i_t} \in G'$  privately to player  $P_i$ ,  $i = 1, \dots, n$ . For instance, this way one can prove that a triple  $(u, v, w) \in G^3$  is a Diffie-Hellman triple with respect to  $g$ , i.e. that  $(u, w) \in \{(g^a, v^a) \mid a \in \mathbb{F}_q\} \subset G \times G$ . Note, in this example,  $\varphi : \mathbb{F}_q \rightarrow G^2$ ,  $a \mapsto (g^a, v^a)$ .

If the order of the group  $H$  is not known, then Shamir’s secret sharing scheme can be replaced by a black box secret sharing scheme [9].

## References

1. Masayuki Abe. Robust distributed multiplication without interaction. In *Advances in Cryptology – CRYPTO ’99*, volume 1666 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
2. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2), 1988.
3. Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2), 1991.
4. Mihir Bellare, and Oded Goldreich. On Defining Proofs of Knowledge. In *Advances in Cryptology – CRYPTO ’92*, volume 740 of *Lecture Notes in Computer Science*. Springer, 1998.

5. Joan F. Boyar, Mark W. Krentel, and Stuart A. Kurtz. A discrete logarithm implementation of zero-knowledge blobs. Technical Report TR-87-02, Department of Computer Science, University of Chicago, 1987.
6. Ronald Cramer and Ivan Damgård. L Linear zero-knowledge: A note on efficient zero-knowledge proofs and arguments. In *29th ACM Symposium on Theory of Computing*. ACM Press, 1997.
7. Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic or: Can zero-knowledge be for free? In *Advances in Cryptology – CRYPTO ’98*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.
8. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO ’94*, volume 839 of *Lecture Notes in Computer Science*. Springer, 1994.
9. Ronald Cramer and Serge Fehr. Optimal black-box secret sharing over arbitrary Abelian groups. In *Advances in Cryptology – CRYPTO ’02*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.
10. Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology – EUROCRYPT ’96*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.
11. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*. IEEE, 1985.
12. Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *17th ACM Symposium on Principles of Distributed Computing*, 1998.
13. Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *16th ACM Symposium on Principles of Distributed Computing*, 1997. Final version appeared in *Journal of Cryptology* 2000.
14. Maurizio Karchmer and Avi Wigderson. On span programs. In *8th Annual Conference on Structure in Complexity Theory*. IEEE, 1993.
15. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT ’99*, *Lecture Notes in Computer Science*. Springer, 1999.
16. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO ’91*, volume 576 of *Lecture Notes in Computer Science*. Springer, 1991.
17. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11), 1979.

## A Examples of $q$ -OWGH Based Commitments

### Based on the DL-Problem

Let  $p$  be prime and  $G = \langle h \rangle$  a subgroup of  $\mathbb{Z}_p^*$  with prime order  $q$ . Then the exponentiation function  $f : \mathbb{Z}_{q-1} \rightarrow G, x \mapsto h^x$  is (a candidate for) a  $q$ -OWGH. Indeed, given  $y \in G$ ,  $v = 0$  fulfils  $f(v) = 1 = y^q$ .

The resulting commitment scheme is the Pedersen commitment scheme we were considering in the first part.

**Based on the RSA-Problem**

The RSA function  $f : \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$ ,  $x \mapsto x^q$  for a prime exponent  $q$  is (a candidate for) a  $q$ -OWGH. Given  $y \in \mathbb{Z}_n^*$ ,  $v = y$  fulfils  $f(v) = y^q$ .

The resulting commitment scheme is  $\text{com}_{g,f}(s, r) = g^s r^q$ .

**Based on Factoring and on the QR-Problem**

Squaring modulo an RSA modulus  $n$ ,  $f : \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$ ,  $x \mapsto x^2$  is (a candidate for) an unconditionally binding 2-OWGH. Given  $y \in \mathbb{Z}_n^*$ ,  $v = y$  fulfils  $f(v) = y^2$  and any quadratic non-residue  $t \in \mathbb{Z}_n^*$  with Jacoby symbol  $+1$  fulfils the requirements for the unconditionally binding property, assumed that the QR-problem is hard.

The resulting computationally binding commitment scheme is  $\text{com}_{g,f}(s, r) = g^s r^2$  for a random quadratic residue  $g$  and the resulting computationally hiding scheme is  $\text{com}_{t,f}(s, r) = t^s r^2$  for a quadratic non-residue  $t$  with Jacoby symbol  $+1$ , both occurring in [2].

**Based on Computing  $n$ -th Roots mod  $n^2$  and on the DCR Assumption**

The function  $f : \mathbb{Z}_{n^2}^* \rightarrow \mathbb{Z}_{n^2}^*$ ,  $x \mapsto x^n$  for an RSA modulus  $n$  is (a candidate for) an unconditionally binding  $n$ -OWGH. Given  $y \in \mathbb{Z}_{n^2}^*$ ,  $v = y$  fulfils  $f(v) = y^n$  and e.g.  $t = \overline{n+1} \in \mathbb{Z}_{n^2}^*$  fulfils the requirements for the unconditionally binding property, based on the decisional composite residuosity (DCR) assumption [15].

The resulting computationally binding commitment scheme is  $\text{com}_{g,f}(s, r) = g^s r^n$  for a random  $n$ -th power  $g$  and the resulting computationally hiding scheme is  $\text{com}_{t,f}(s, r) = t^s r^n$  for e.g.  $t = \overline{n+1} \in \mathbb{Z}_{n^2}^*$ , i.e. the Paillier encryption function [15].

Note that even though  $n$  is not a prime, it can be treated in this context as one, as it is (assumed to be) hard to find non-trivial divisors.

# Asynchronous Secure Communication Tolerating Mixed Adversaries

K. Srinathan<sup>1,\*</sup>, M.V.N. Ashwin Kumar<sup>2</sup>, and C. Pandu Rangan<sup>1,\*\*</sup>

<sup>1</sup> Department of Computer Science and Engineering, Indian Institute of Technology,  
Madras, Chennai-600036, India,

`ksrinath@cs.iitm.ernet.in, ranganc@iitm.ernet.in`

<sup>2</sup> Department of Computer Science, Cornell University,  
Ithaca, New York,

`mvnak@cs.cornell.edu`

**Abstract.** We study the problem of secure communication tolerating generalized mixed adversaries across an underlying completely asynchronous incomplete network. We explore the interplay between the minimal network connectivity required and the degree of security attainable, and completely characterize the network requirements for attaining perfect and unconditional (with negligible error) security. We also consider networks with additional broadcast capabilities and prove that unconditionally secure communication can be achieved with much lesser connectivity if the network assures the broadcast primitive.

## 1 Introduction

Consider  $n$  players who are connected by an underlying communication network  $\mathcal{N}$ . Our concern is to make sure that every player can *talk* to every other player. Two players can *talk* to each other if they are connected by an edge. Hence, we can trivially guarantee that every player can *talk* to every other player if the underlying network  $\mathcal{N}$  is complete. But do we require all the  $nC_2$  direct connections (or edges)? Can we not ensure that all the players can communicate with one another with a lesser number of edges? Evidently, the smallest connected network (viz. a tree) would suffice to allow every pair of players to be able to *talk* (though indirectly). However, such minimal connectivity is not enough if one player wants to *secretly talk* to another player, i.e., the sender  $\mathbf{S}$  has to transmit a message to the receiver  $\mathbf{R}$  such that all the other players should get no information about the message transmitted, even if some non-trivial subset of players (excluding  $\mathbf{S}$  and  $\mathbf{R}$ ) collude and behave maliciously. The interplay between information-theoretically secure communication and minimal network connectivity has been studied extensively. Dolev et. al. in [5] proved that a *synchronous* network has to be at least  $(\max(t_a, t_p) + t_a + 1)$ -connected for secure

---

\* Financial support from Infosys Technologies Limited, India, is acknowledged.

\*\* Partially supported by DRDO collaborative project on Communication and Networking Technologies.

message transmission to be guaranteed between every two players, where up to  $t_p$  players collude and only eavesdrop on the messages routed through them (passive faults), while up to  $t_a$  players collude and maliciously try to disrupt the protocol apart from eavesdropping (active or Byzantine faults). These result were recently generalized in [12] to the non-threshold case modeling only Byzantine faults: perfectly secure transmission is possible if and only if the union of the players in no two potentially faulty subsets of players forms a vertex cut-set of the graph (wherein the potentially faulty subsets of players are enumerated as an *adversary structure* [9]). However, these results are restricted to the case where the underlying network is synchronous. Asynchronous perfectly secure communication for the case of threshold adversaries was studied in [13]. In essence, a  $(\max(t_a, t_p) + 2t_a + 1)$ -connected network is necessary and sufficient. Unconditionally secure communication with negligible error in reliability (i.e., **R** might receive a wrong message) was studied in detail in [8]. However, these results are restricted to the threshold case assuming that the underlying network is synchronous. We initiate a study of asynchronous secure communication tolerating faulty players, some passive and some others active, characterized as generalized mixed adversary structures (as in [7]) and investigate the minimal connectivity requirements for perfect security and unconditional security.

A very important primitive used by all protocols for secure communication is that of *reliable, yet insecure* communication (e.g. [5] calls it public transmission). By this we mean, any message  $m$  sent by player **S** to player **R** is *correctly* received by **R**; however, the other players may have considerable (or even full) information about  $m$ . We can achieve this *reliable* communication trivially if we are assured that the network  $\mathcal{N}$  has *broadcast* capability.<sup>1</sup> However, if  $\mathcal{N}$  does not have broadcast capabilities, one should be able to simulate reliable transmission using a protocol. We study the minimum network connectivity which guarantees the possibility of such reliable transmission in networks without broadcast capabilities. We show that the existence of a broadcast channel does *not* reduce the connectivity requirement for perfectly secure asynchronous communication.

In line with [7], the generalized mixed adversary is characterized by a generalized adversary structure (see Definition II), i.e. a set of pairs  $(D, E)$ , where  $D$  and  $E$  are disjoint subsets of the set of players, wherein the adversary may select one arbitrary pair from the structure and corrupt the players in  $D$  actively (i.e. take full control) and in addition passively corrupt (i.e. read and process information of) the players in  $E$ . Among our results, we show that in the perfect setting, secure message transmission between any pair of (honest) nodes in a completely asynchronous network is possible if and only if neither the removal of the players in the union of any *three* sets of potential active collusions, nor the removal of the players in the union of any *two* sets of potential active collusions with any *one* corresponding set of potential passive collusion, leaves the network *disconnected*. Evidently, the above condition generalizes the threshold adversary requirement of  $(\max(t_a, t_p) + 2t_a + 1)$ -connected network. Interestingly, we prove

<sup>1</sup> By definition, if a message  $m$  is sent using a *broadcast* channel then *all* the players correctly receive (the *same*)  $m$ .

that the same condition given above is necessary and sufficient even in the case of unconditional security, though the resultant protocol is less complex. Thus, the minimal connectivity requirement is unaffected by the weakening of security. However, in the presence of a broadcast channel, the perfect setting *still* requires the same amount of connectivity whereas in the unconditional setting, it is (necessary and) sufficient if the players in the union of any *two* sets of potential active collusions with any *one* corresponding set of potential passive do not form a *vertex cut-set* of the network (i.e., only the second half of the above condition is sufficient). In all the above cases, the designed protocols have both their computation and communication complexities polynomial in the size of the maximal basis of the adversary structure.

Motivated by the facts that network synchrony is hard to achieve and that a threshold adversarial model is insufficient to model all types of mutual (dis)trust, we generalize the results of [5,13,12] to the generalized mixed adversary model and/or to asynchronous networks (see Theorem 7), in the perfect setting. Furthermore, in the unconditional setting, we initiate the study of asynchronous secure communication (see Theorem 5) and the study of unconditional with broadcast capability model (see Theorem 3).

Asynchronous secure communication is an important primitive for secure multiparty computation over asynchronous incomplete networks. Thus, our results can be used to transform the asynchronous secure computation protocols that run over a complete network (e.g. [2,3]) into ones that can be executed over incomplete networks.

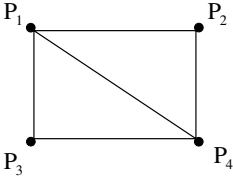


Fig. 1. Network.

The usefulness of some of our results is illustrated, for instance, through the following implication: Consider a asynchronous chorded ring network of four players as shown in Fig. 1. The most powerful adversary that previous known protocols (e.g. [13]) for perfectly secure communication among the players over the asynchronous network in Fig. 1 could tolerate is one that *passively* corrupts *one* arbitrary player (since the chordal ring network is 2-connected, we require that  $2 > \max(t_a, t_p) + 2t_a$ , giving  $t_a = 0$  and  $t_p = 1$ ). Using

our results, one can perfectly tolerate an adversary that passively corrupts player  $P_1$  or player  $P_4$  or (even) *actively* corrupts player  $P_2$  or player  $P_3$ .

## 2 Preliminaries

We consider a network  $\mathcal{N}(\mathcal{P}, \mathcal{E})$ , where  $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$  denote the set of players (nodes) in the network that are connected by the edges as defined by  $\mathcal{E} \subseteq \mathcal{P} \times \mathcal{P}$ . Formally, all the  $n$  players (nodes) in the network  $\mathcal{N}$  can be modeled as probabilistic interactive Turing Machines. We assume that randomization is achieved through random bits. We assume that the underlying network  $\mathcal{N}$  is *asynchronous*, i.e., a message sent on a channel/path can be arbitrarily delayed (similar to the communication model in [6]). However, if two nodes  $P_i$  and  $P_j$

are directly connected by a link (edge), then each message that player  $P_i$  sends to  $P_j$  through the link is eventually received (albeit probably in any order).

The set of faults in the players is usually captured using the notion of an external centralized *adversary*. A computationally unbounded *adversary*  $\mathcal{B}$  is a probabilistic strategy that controls/corrupts a subset of players (and/or the edges connecting the players) and endeavors to violate the security of the system. We assume, without loss of generality, that the adversary can control/corrupt only the players and *not* the edges connecting them.<sup>2</sup>

Our notional adversary is a *passive* adversary if all dishonest players exhibit only *passive* adversarial behavior, that is, all the corrupted players can collusively gather all the information they get and run any arbitrary computation on this information. The adversary is a *Byzantine* adversary if all dishonest players show *active* adversarial behaviour, that is, in addition to *eavesdropping*, they can maliciously alter their behavior in an arbitrary and coordinated fashion.<sup>3</sup> If some players exhibit only *passive* adversarial behavior while others exhibit *active* adversarial behavior, the adversary is called a *mixed* adversary. Depending upon the amount of knowledge one has about the adversarial behaviour of the players, adversaries can be modeled as either *threshold* adversaries or *generalized* (or *non-threshold*) adversaries. When modeled as a *threshold* adversary, a maximum of  $t$  out of the  $n$  players are assumed to exhibit adversarial behaviour. Hirt and Maurer [9] transferred and adjusted the notion of *access structures* (introduced in [10] for secret sharing) to the field of general secure multiparty computation, which was subsequently adapted to the secure communication setting in [12]: the behaviour of the faulty players is characterized by an *adversary structure*, which is a monotone set of subsets of players, wherein the players in any *one* of these subsets is corruptible by the adversary.

In our study we consider *mixed* adversaries modeled using generalized adversary structures (like in [7]). In this model, some subset of players  $D$  show *active* adversarial behaviour and *at the same time*, some other subset of players  $E$  show only *passive* adversarial behaviour. Hence, the adversary is characterized by a monotone<sup>4</sup> set of classes  $C = (D, E)$ , where  $D, E \subset \mathcal{P}$  and  $D \cap E = \emptyset$ . The players in *one* specific class is corruptible by the adversary. – players in  $D$  are actively corrupted while those in  $E$  are passively corrupted.

**Definition 1** ([7]). A generalized mixed adversary structure  $\mathcal{A}$  is a monotone set of classes  $C = (D, E)$ , where  $D, E \subset \mathcal{P}$  and  $D \cap E = \emptyset$ . The maximal basis of  $\mathcal{A}$  is defined as the collection of classes  $\{(D, E) | (D, E) \in \mathcal{A}, \nexists (X, Y) \in \mathcal{A}, ((X \supset D) \cap (Y \supset E))\}$ . We abuse the notation  $\mathcal{A}$  to denote the maximal basis.

<sup>2</sup> This is because, any adversary corrupting both the players and edges of a network  $\mathcal{N}$  can be simulated by an adversary corrupting the players alone on a new network  $\mathcal{N}'$  got by replacing each insecure edge  $e = (P_i, P_j)$  by a player  $P_{ij}$  and two edges  $e_1 = (P_i, P_{ij})$  and  $e_2 = (P_{ij}, P_j)$ .

<sup>3</sup> Note that this subsumes fail-stop faults wherein the dishonest players alter their behaviour in a pre-specified manner, viz., do not respond at all.

<sup>4</sup> *Monotone* means that if a class  $C = (D, E)$  belongs to the structure, then all classes  $C' = (D', E')$  such that  $D' \subseteq D$  and  $E' \subseteq E$  are also elements of the structure.

**Remark:** Similar to the classical threshold model, the threshold mixed adversarial model can be defined as one in which up to  $t_a$  players maliciously attempt to disrupt the protocol, while up to  $t_p$  other players only eavesdrop.

We introduce the notion of a *path adversary* (like in [12]). Any message transmitted from a player  $P_i$  to a player  $P_j$  should traverse a path in  $\mathcal{N}$  connecting the players. Hence, it is more appropriate to consider paths as corruptible entities rather than considering the adversarial behaviour of the individual players. This *path adversary* we characterize with the help of a generalized mixed structure. Let the set of all paths between players  $P_i$  and  $P_j$  in  $\mathcal{N}$  be denoted by  $\mathcal{X}_{path}(P_i, P_j)$ .

**Definition 2.** Given the generalized mixed adversary structure  $\mathcal{A}$ , we denote the path adversary structure over a subset of paths  $\Phi(P_i, P_j) \subseteq \mathcal{X}_{path}(P_i, P_j)$  as  $\mathcal{A}_{path}^{[\Phi]}(P_i, P_j)$ .  $\mathcal{A}_{path}^{[\Phi]}(P_i, P_j)$  is a monotone set of subset pairs of  $\Phi(P_i, P_j)$ . For every class  $C = (D, E) \in \mathcal{A}$  there is a corresponding class  $\Lambda = (\Lambda_D, \Lambda_E) \in \mathcal{A}_{path}^{[\Phi]}(P_i, P_j)$  such that  $\Lambda_D$  (or  $\Lambda_E$ ) is the set of all paths in  $\Phi(P_i, P_j)$  between  $P_i$  and  $P_j$  passing through any of the players in  $D$  (or  $E$ , respectively). More precisely,  $\mathcal{A}_{path}^{[\Phi]}(P_i, P_j) \subset 2^{\Phi(P_i, P_j)}$ , such that

$$\mathcal{A}_{path}^{[\Phi]}(P_i, P_j) = \{(\Lambda_D, \Lambda_E) \mid \forall (D, E) \in \mathcal{A}, (\Lambda_D = \Phi(P_i, P_j) \setminus (N_{[P \setminus D]})) \cap (\Lambda_E = \Phi(P_i, P_j) \setminus (N_{[P \setminus E]}))\}$$

where  $N_{[V]}$  denotes the set of all paths in the sub-network induced by  $\mathcal{N}$  on the vertices in  $V$ .

**Definition 3.** Given the generalized mixed adversary structure  $\mathcal{A}$ , the network is said to be  $\mathcal{A}^{(k, \ell)}$ -connected if for any  $\max(k, \ell)$  classes  $C_{i_1}, C_{i_2}, \dots, C_{i_{\max(k, \ell)}}$  from  $\mathcal{A}$ , the deletion of the nodes in  $\bigcup_{j=1}^k D_{i_j} \cup \bigcup_{j=1}^\ell E_{i_j}$  from the network does not disconnect the network. With respect to two players (nodes)  $P_i$  and  $P_j$ , the network is said to be  $\mathcal{A}^{(k, \ell)}(P_i, P_j)$ -subconnected if for any  $\max(k, \ell)$  classes  $C_{i_1}, C_{i_2}, \dots, C_{i_{\max(k, \ell)}}$  from  $\mathcal{A}$ , the deletion of the nodes in  $\bigcup_{j=1}^k D_{i_j} \cup \bigcup_{j=1}^\ell E_{i_j}$  from the network does not render  $P_i$  unreachable from  $P_j$ .

**Remark:** It is evident that for the threshold mixed adversarial model, the  $\mathcal{A}^{(k, \ell)}$ -connectivity condition translates to  $\kappa > kt_a + \ell t_p$ , where  $\kappa$  denotes the size of the smallest vertex cut.

Since the underlying network is asynchronous, the adversary has the power to *schedule* the messages. A message routed on a path having an actively corrupted player (which we shall call henceforth as an actively corrupted path) can schedule the message in such a way that the receiver will be made to wait for it for arbitrary long periods of time. Actually, these actively corrupted paths may just withhold the messages routed through them and thus receiver **R** may not listen from the sender **S** on paths in  $\Lambda_D$ ,  $(\Lambda_D, \Lambda_E) \in \mathcal{A}_{path}^{\mathcal{X}}(\mathbf{S}, \mathbf{R})$ . However, the receiver can not distinguish between *honest* paths which are *slow* (thanks to the malicious scheduling) and *malicious* paths which *withhold* information. Hence, in the worst case, the set of paths on which **R** can expect to receive information might contain all the *malicious* and the *eavesdropping* paths!



### 3 Secure Message Transmission

We consider the problem of transmitting a message  $m$  from a player  $P_i$  to a player  $P_j$  *securely*. We consider perfect and unconditional (with a small error probability) security in the information theoretic sense. In this section we define perfect security and unconditional security. Let the message to be transmitted securely be drawn from a (prespecified) fixed finite field  $\mathcal{F}$  and let  $\Gamma$  denote the underlying probability distribution on this field. Define the VIEW of a player  $P_j$  in  $\mathcal{N}$ , at any point of the execution of a protocol  $\Pi$  for secure message transmission, to be the information the player can get from its *local input* to the protocol (if any), all the messages that  $P_j$  had earlier *sent* or *received*, the protocol *code* executed by  $P_j$  and its *random coins*. The VIEW of the adversary (i.e. the VIEW of the players exhibiting adversarial behaviour) at any point of the execution of  $\Pi$  is defined as all the information that the adversary can get from the VIEWS of all the players corrupted by the adversary (i.e. all the information that these players can commonly compute from their VIEWS).

For every message  $m \in \mathcal{F}$ , any adversary  $\mathcal{B}$  characterized by  $\mathcal{A}$ , and any protocol  $\Pi$  for secure message transmission, let  $\hat{\Gamma}(\mathcal{B}, m, \Pi)$  denote the probability distribution on the VIEW of the adversary  $\mathcal{B}$  at the end of the execution of  $\Pi$  when the message sent is  $m$ .

**Definition 4 (Secure Message Transmission).** *A protocol  $\Pi$  is said to facilitate perfectly secure message transmission between two players  $P_i$  and  $P_j$  if for any message  $m$ , drawn from  $\Gamma$  on  $\mathcal{F}$ , and for every adversary  $\mathcal{B}$ , characterized as an (generalized mixed) adversary structure  $\mathcal{A}$ , the following conditions are satisfied:*

1. *Secrecy:  $\hat{\Gamma}(\mathcal{B}, m', \Pi) \equiv \hat{\Gamma}(\mathcal{B}, m, \Pi) \quad \forall m' \in \mathcal{F}$ . That is, the above two distributions are identical irrespective of the message transmitted.*
2. *Resiliency: The protocol certainly terminates with the receiver  $P_j$  receiving the message  $m$  correctly.*

*The protocol  $\Pi$  is said to be unconditionally secure (with negligible error) if a negligible error probability  $\delta$  can be tolerated with respect to the Resiliency condition, i.e., the protocol terminates with an overwhelming probability  $1 - \delta$  and the receiver  $P_j$  receives  $m$  with a negligibly small error probability  $\delta$ . The probability is over the choice of  $m$  and the coin flips of each of the players and the adversary (this is same as the  $(0, \delta)$ -security as defined in [8]).*

### 4 Issues

Before we start designing protocols for asynchronous secure communication between the sender  $\mathbf{S}$  and the receiver  $\mathbf{R}$  over the network  $\mathcal{N}$ , tolerating generalized mixed adversaries, there are a few critical issues which have to be dealt with:

1. *What are the paths that, out of the potentially exponential number of paths between  $\mathbf{S}$  and  $\mathbf{R}$ , should be used for transmission?* Note that irrespective

of the total number of paths from  $\mathbf{S}$  to  $\mathbf{R}$ , only a polynomial (on the input size, i.e.,  $n + |\mathcal{A}|$ ) sized subset of the paths should be used if the resultant protocol is to be feasible. Furthermore, one should be able to compute the above subset of paths in polynomial time!

2. *What to send along the above chosen paths?* In the sequel, it is shown that the answer depends on the setting.
3. *How to route a message along a path?* Since, the adversary can actively corrupt some players, these intermediate players can misroute a message.
4. *How does the receiver  $\mathbf{R}$  distinguish between two different paths having the same final link?* This may be required since the receiver invariably has to “reconstruct” the sender’s message from the data that he receives via many different paths, and the data may be an ordered set.

#### 4.1 Solving Issue #1: Critical Paths

Consider a network in which at most  $t$  nodes are faulty. In such a case, irrespective of which nodes are corrupted it is evident that in the worst case, not more than  $t$  disjoint paths can be corrupted (one node per path). Hence if the network is  $\kappa$ -connected, it is sufficient to abstract the network as  $k$  disjoint paths between  $\mathbf{S}$  and  $\mathbf{R}$  of which any  $t$  paths could be faulty; or even better as  $k$  wires of which any  $t$  could be corrupted. This is exactly what [5,13] have done! In the above case, we call the  $k$  (disjoint) paths that are chosen as the *critical paths*. Note that the number of critical paths is usually much lower than the total number of paths between  $\mathbf{S}$  and  $\mathbf{R}$ . Unfortunately, when the players’ adversarial behaviour is modeled as a generalized mixed adversary, however, the non-disjointness of the communication paths is *indispensable*.

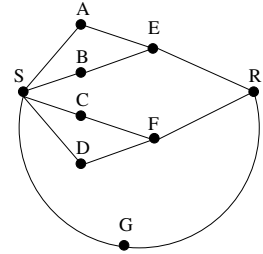


Fig. 2. Network  $\mathcal{N}_1$

For example, consider the network  $\mathcal{N}_1$  in the Fig. 2. Let the adversary be characterized by the following mixed adversary structure  $\mathcal{A} = \{(A, \emptyset), (B, \emptyset), (F, \emptyset), (G, \emptyset)\}$ . There are in total five paths from  $\mathbf{S}$  to  $\mathbf{R}$ . It can be easily seen (using the results of [5,13]) that any protocol for asynchronous secure message transmission between  $\mathbf{S}$  and  $\mathbf{R}$  should necessarily use four of the paths between  $\mathbf{S}$  and  $\mathbf{R}$ , leaving out one of the two paths passing through  $\mathbf{F}$  (since the node  $\mathbf{F}$  is potentially corruptible). Note that in any case, the chosen four paths are all not disjoint! Furthermore, the path that is left out is not a critical path.

We now need to develop a deterministic methodology for computing the critical paths; moreover, we require that the algorithm runs in time polynomial in the input size. Assume that the sender  $\mathbf{S}$  and receiver  $\mathbf{R}$  are  $\mathcal{A}^{(k_1, 0)}$ -subconnected as well as  $\mathcal{A}^{(k_2, 1)}$ -subconnected.<sup>5</sup> We solve the above issue by providing an al-

<sup>5</sup> It will be clear from the sequel that in the various settings considered in this paper, we will be dealing with only  $\mathcal{A}^{(3, 0)}$ -subconnectivity and  $\mathcal{A}^{(2, 1)}$ -subconnectivity. More precisely, in the perfect and unconditional settings,  $k_1 = 3$  and  $k_2 = 2$ ; and in the unconditional with broadcast setting,  $k_1 = 0$  and  $k_2 = 2$ .

**Computing  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$** 

**Inputs:**  $\mathcal{N}(\mathcal{P}, \mathcal{E})$ ,  $\mathcal{A}$ , sender  $\mathbf{S}$  and receiver  $\mathbf{R}$ .

Let  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) = \emptyset$  and let  $\mathcal{A}_{access}^{(1)} = \{A_i = (\mathcal{P} \setminus D) \mid \forall (D, E) \in \mathcal{A}\}$ .

For  $i_1 = 1$  to  $|\mathcal{A}_{access}^{(1)}|$

    For  $i_2 = i_1 + 1$  to  $|\mathcal{A}_{access}^{(1)}|$

    ...

        For  $i_{k_1} = i_{k_1-1} + 1$  to  $|\mathcal{A}_{access}^{(1)}|$

            IF  $(Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) \cap \mathcal{N}_{[A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_{k_1}}]}) = \emptyset$

                THEN Select at random some path  $p$  in  $\mathcal{N}_{[A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_{k_1}}]}$

                    and set  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) \leftarrow Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) \cup \{p\}$ .

            NEXT  $i_{k_1}$

        ...

        NEXT  $i_2$

    NEXT  $i_1$

Let  $\mathcal{A}_{access}^{(2)} = \{A_i = (\mathcal{P} \setminus (D \cup E)) \mid \forall (D, E) \in \mathcal{A}\}$ .

For  $i_1 = 1$  to  $|\mathcal{A}_{access}^{(2)}|$

    ...

        For  $i_{k_2} = i_{k_2-1} + 1$  to  $|\mathcal{A}_{access}^{(2)}|$

            IF  $(Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) \cap \mathcal{N}_{[A_{i_1} \cap \dots \cap A_{i_{k_2}}]}) = \emptyset$

                THEN Select at random some path  $p$  in  $\mathcal{N}_{[A_{i_1} \cap \dots \cap A_{i_{k_2}}]}$

                    and set  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) \leftarrow Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R}) \cup \{p\}$ .

            NEXT  $i_{k_2}$

        ...

    NEXT  $i_1$

**Comment:** The above construction ensures that  $\mathbf{S}$  and  $\mathbf{R}$  are  $\mathcal{A}^{(k_1, 0)}$ -subconnected as well as  $\mathcal{A}^{(k_2, 1)}$ -subconnected even if the set of paths is restricted to  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$ .

**Fig. 3.** Identifying the critical paths  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$ .

gorithm (see Fig. 3) with the following properties: (1) The algorithm takes as input  $\mathcal{N}(\mathcal{P}, \mathcal{E})$ ,  $\mathcal{A}$ , and the sender  $\mathbf{S}$  and the receiver  $\mathbf{R}$ . (2) The algorithm outputs a set of paths between  $\mathbf{S}$  and  $\mathbf{R}$  in  $\mathcal{N}$ , denoted by  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$ . (3) The algorithm runs in time polynomial in  $|\mathcal{P}|$  and  $|\mathcal{A}|$ , viz.  $O(|\mathcal{P}| \cdot |\mathcal{A}|^6)$ . (4) The number of paths in  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$  is polynomial in  $|\mathcal{A}|$ , viz.  $O(|\mathcal{A}|^3)$ . (5) A solution using  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$  exists if and only if a solution that uses the full set of paths  $\mathcal{X}_{path}(\mathbf{S}, \mathbf{R})$  exists, i.e. it is ensured that  $\mathbf{S}$  and  $\mathbf{R}$  are  $\mathcal{A}^{(k_1, 0)}$ -subconnected as well as  $\mathcal{A}^{(k_2, 1)}$ -subconnected even if the set of paths is restricted to  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$ .

**Theorem 1.** The algorithm in Fig. 3 satisfies all the above stated properties. We assume the worst case of  $k_1 = 3$  and  $k_2 = 2$ .

**PROOF OF PROPERTY 3:** The only computational intensive step is the IF step, which takes  $O(|\mathcal{P}| \cdot |Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})|)$  time. Since,  $|Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})| = O|\mathcal{A}|^3$  (see Proof of Property 4), the overall computational complexity is  $O(|\mathcal{P}| \cdot |\mathcal{A}|^6)$ .

**PROOF OF PROPERTY 4:** The property is clear from the fact that in each of the  $\left\{ \binom{|\mathcal{A}|}{3} + \binom{|\mathcal{A}|}{2} \right\}$  iterations, the size of  $Pot\mathcal{X}_a(\mathbf{S}, \mathbf{R})$  increases by at most one.

**PROOF OF PROPERTY 5:** Follows from the construction. □

## 4.2 Solving Issues #2 & #4: Anonymous Secret Sharing

We require that the adversary should get no information about the message, whilst  $\mathbf{R}$  should be able to reconstruct the message (or should at least be able to detect a fault, if not correct it!). This is reminiscent of secret sharing. Moreover,  $\mathbf{R}$  may not be able to distinguish between shares routed on *different* paths arriving through the *same* final link. Therefore the requirement is that of *anonymous secret sharing* [4]. In our case, anonymization is easily achieved by creating *self-identifying* message packets by appending the intended path number to the message, i.e., if the message packet  $\varpi_i$  is to be routed through path  $p_i$ , the *self-identifying* packet would be  $(\varpi_i, i)$ . The above abstraction helps us work with secret sharing alone since it can easily be “compiled” into anonymous secret sharing. In the unconditional case, as will be illustrated in the sequel, it is sufficient if the secret is split into shares such that their sum gives back the secret. However, in the perfect setting, we use the linear perfect secret sharing schemes based on monotone span programs [11].

## 4.3 Solving Issue #3: Routing Algorithm

As a recap, a routing primitive is essential since it is not guaranteed that a message intended to be routed along a path will reach the receiver on that particular path. We prove that such a strong primitive is not required and the weaker primitive described in Observation [11] is sufficient to design secure message transmission protocols. We next provide the routing primitive  $\Delta_{async}$  (see Fig. [4]) for asynchronous networks.

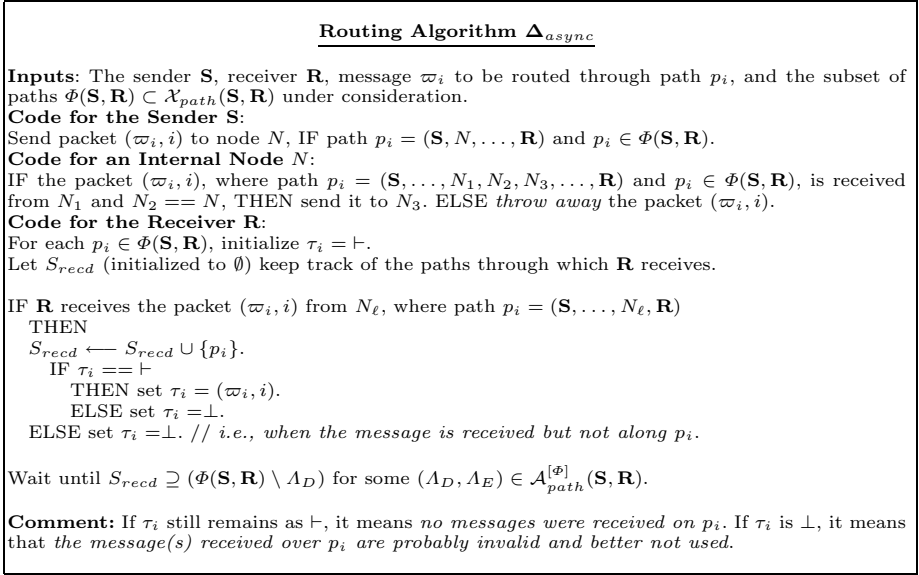
**Observation 11.** *It is sufficient to have a routing algorithm  $\Delta$  that guarantees that on every honest path  $p_i$  (identified by  $i$ ),  $\mathbf{R}$  receives exactly one correct self-identifying packet, namely  $(\varpi_i, i)$ .*

We now prove that the algorithm  $\Delta_{async}$  satisfies the weak routing primitive described in Observation [11].

**Theorem 2.** *The Routing Algorithm  $\Delta_{async}$  for asynchronous networks, given in Fig. [4], satisfies the specification as in Observation [11].*

PROOF: Let  $p_i$  be an *honest* path through which  $\mathbf{S}$  intended to send message  $\varpi_i$ . On the contrary, assume that  $\mathbf{R}$  received on path  $p_i$  either:

1. *One incorrect packet:* This leads to a contradiction because if the packet was correct up to  $q$  hops, our algorithm ensures that it is correct even after  $q + 1$  hops. Proof follows through induction.
2. *No packet:* As all the nodes on the path are honest it is clear that at least one packet (viz. the packet routed through that path  $(\varpi_i, i)$ ) will *eventually* reach  $\mathbf{R}$ . (Note that, however,  $\mathbf{R}$  may not wait for this message.)
3. *More than one packet:* Since more than one packet was received on the honest path  $p_i$  (with the same path identifier), there exists at least one packet whose path identifier was corrupted to  $i$ . Let  $N$  be the corrupted node where the



**Fig. 4.** The Routing Algorithm for Asynchronous Networks.

path identifier was corrupted to  $i$  and sent to a honest node  $N_h$  in  $p_i$ .  $\Delta_{async}$  ensures that this packet is *thrown away* by  $N_h$ . Notice that the same holds even when  $N_h = \mathbf{R}$ .  $\square$

## 5 Unconditionally Secure Communication with Broadcast

### 5.1 Impossibility

**Theorem 3.** *Unconditionally secure message transmission tolerating  $\mathcal{A}$  across an asynchronous network  $\mathcal{N}$  with broadcast capability is possible only if the sender  $\mathbf{S}$  and the receiver  $\mathbf{R}$  are  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected.*

PROOF: Assume that, on the contrary, secure transmission with negligible error is possible even when  $\mathbf{S}$  and  $\mathbf{R}$  are *not*  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected. In this case, the adversary can exploit the asynchrony of the network and delay the messages routed through the paths in  $\Lambda_D$  for some class  $\Lambda = (\Lambda_D, \Lambda_E) \in \mathcal{A}_{path}^{\mathcal{X}}(\mathbf{S}, \mathbf{R})$ . Since  $\mathbf{S}$  and  $\mathbf{R}$  are not  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected, there exists a class  $\Lambda'$  such that  $\Lambda' = (\Lambda'_D, \Lambda'_E) \in \mathcal{A}_{path}^{\mathcal{X}}(\mathbf{S}, \mathbf{R})$  such that  $\Lambda_D \cup \Lambda'_D \cup \Lambda'_E = \mathcal{X}_{path}(\mathbf{S}, \mathbf{R})$ . Thus by choosing to corrupt this class  $\Lambda'$  the adversary has all the knowledge that  $\mathbf{R}$  takes into consideration thus violating the secrecy requirement.  $\square$

### 5.2 Possibility

We propose a protocol sketch for unconditionally secure communication on the lines of [8] and show that the  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -connectivity condition is sufficient.

**Asynchronous Unconditionally Secure Transmission**

1. **S** sends different  $\rho_j^{\mathbf{S}}, \sigma_j^{\mathbf{S}} \in \mathcal{F}$  on each path  $p_j$  in  $\Phi(\mathbf{S}, \mathbf{R})$ .
2. For each,  $\rho_j^{\mathbf{R}}, \sigma_j^{\mathbf{R}}$  that **R** receives on the correct path  $p_j$ , **R** *reliably sends* (using the broadcast channel) a random  $\nu_j^{\mathbf{R}} \in \mathcal{F}$  and  $s_j^{\mathbf{R}} = (\nu_j^{\mathbf{R}} \cdot \rho_j^{\mathbf{R}} + \sigma_j^{\mathbf{R}})$  to **S**.
3. **S** constructs  $G = \{j | s_j^{\mathbf{R}} = (\nu_j^{\mathbf{R}} \cdot \rho_j^{\mathbf{S}} + \sigma_j^{\mathbf{S}})\}$ . **S** *reliably sends* (using the broadcast channel)  $G$  and  $Z = m^{\mathbf{S}} + \sum_G \rho_j^{\mathbf{S}}$ .
4. **R** computes  $m^{\mathbf{R}} = Z - \sum_G \rho_j^{\mathbf{R}}$ .

**Fig. 5.** Unconditionally Secure Transmission with Broadcast.

**Theorem 4.** *The protocol given in Fig. 5 is indeed a protocol guaranteeing unconditionally secure communication if the sender **S** and the receiver **R** are  $\mathcal{A}^{(2,1)}$ -subconnected.*

**PROOF OF SECRECY:** Since the network is asynchronous, the receiver can expect to receive messages only on paths in  $(\mathcal{X}_{path}(\mathbf{S}, \mathbf{R}) \setminus \Lambda_D)$  (for some  $\Lambda = (\Lambda_D, \Lambda_E) \in \mathcal{A}_{path}^{\mathcal{X}}(\mathbf{S}, \mathbf{R})$ ). However, since **S** and **R** are  $\mathcal{A}^{(2,1)}$ -subconnected, even if the adversary corrupts some other class  $\Lambda' = (\Lambda'_D, \Lambda'_E)$ , there exists one path  $p_h$  on which the messages will reach **R** and has no corrupted (active or passive) player. Hence, **R** receives the values  $\rho_j^{\mathbf{S}}$  and  $\sigma_j^{\mathbf{S}}$  correctly. The adversary can not find out the value  $\rho_j^{\mathbf{S}}$  even using  $s_j^{\mathbf{R}}$  and  $\nu_j^{\mathbf{R}}$ . Hence the adversary can only *guess* the value of  $m^{\mathbf{S}}$  even after knowing  $Z$ .

**PROOF OF RESILIENCY:**  $m^{\mathbf{S}} \neq m^{\mathbf{R}}$  if and only if  $\rho_j^{\mathbf{S}} \neq \rho_j^{\mathbf{R}}$  for some  $j \in G$ . This occurs with a probability  $\frac{1}{|\mathcal{F}|}$ . Hence,  $Pr(m^{\mathbf{S}} \neq m^{\mathbf{R}}) \leq \frac{|G|}{|\mathcal{F}|}$ , which can be made sufficiently small since one could choose the working field such that  $|\mathcal{F}| > \frac{|G|}{\delta}$  and still have the compute, round and communication complexity of the resultant protocol polynomial in the size of the network, the size of the maximal basis of the adversary structure and  $\log \frac{1}{\delta}$  (if  $\delta > 0$ ).  $\square$

## 6 Unconditionally Secure Communication without Broadcast

### 6.1 Impossibility

**Theorem 5.** *Unconditionally secure message transmission tolerating  $\mathcal{A}$  across an asynchronous network  $\mathcal{N}$  without broadcast capability is possible only if the sender **S** and the receiver **R** are  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected as well as  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected.*

**PROOF:** As a direct consequence of Theorem 3, **S** and **R** should be at least  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected for any secure communication protocol to satisfy the *secrecy* condition. For the sake of contradiction, assume that there exists a scheme  $\xi$  for unconditionally secure communication even when **S** and **R** are *not*

$\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected. Using  $\xi$ , we construct a protocol  $\xi'$  for unconditionally secure communication between a sender  $\mathbf{S}'$  and a receiver  $\mathbf{R}'$  over a network  $\mathcal{N}'$ , wherein  $\mathbf{S}'$  and  $\mathbf{R}'$  are connected by exactly *three* disjoint paths ( $p_1, p_2$  and  $p_3$ ), at least one of which is corrupted by a Byzantine adversary. However, in this case, secure communication of any sort is impossible. The adversary can delay the message through one of the paths so that  $\mathbf{R}'$  does not take it into consideration. Then the adversary can corrupt one of the other two messages. Thus  $\mathbf{R}'$  will have to reconstruct the secret using one of the two messages and in a sense has exactly the same amount of information as the adversary. Since the adversary should not get more information about the secret (than what he can get by random guessing), unconditionally secure communication is not possible.

Construction of  $\xi'$  from  $\xi$  is simple. Since,  $\mathbf{S}$  and  $\mathbf{R}$  are not  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected, there exist three classes  $\Lambda_1 = (\Lambda_{D_1}, \Lambda_{E_1})$ ,  $\Lambda_2 = (\Lambda_{D_2}, \Lambda_{E_2})$ ,  $\Lambda_3 = (\Lambda_{D_3}, \Lambda_{E_3})$  in the path adversary structure such that  $\Lambda_{D_1} \cup \Lambda_{D_2} \cup \Lambda_{D_3} = \mathcal{X}_{path}(\mathbf{S}, \mathbf{R})$ . Construct scheme  $\xi'$  wherein, every message sent through a path in  $\Lambda_{D_1}$  in  $\xi$  is sent through the path  $p_1$  in  $\xi'$ ; every message sent through a path in  $\Lambda_{D_2} \setminus \Lambda_{D_1}$  in  $\xi$  is sent through the path  $p_2$  in  $\xi'$ ; and every message sent through a path in  $\Lambda_{D_3} \setminus (\Lambda_{D_1} \cup \Lambda_{D_2})$  in  $\xi$  is sent through the path  $p_3$  in  $\xi'$ . Hence, if  $\xi$  is a protocol for unconditionally secure communication, so is  $\xi'$ .  $\square$

## 6.2 Possibility

We remark that the protocol for unconditionally secure communication in Fig. 5 will work even in this case, if we are able to simulate *reliable but insecure* transmission. We provide below a protocol for the same (which we call PUBLIC transmission) whenever the sender and receiver are at least  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected.

**Theorem 6.** *Reliable, yet insecure transmission is possible if and only if  $\mathbf{S}$  and  $\mathbf{R}$  are  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected.*

NECESSARY: Assume the contrary. Then there exists classes  $\Lambda_1, \Lambda_2$  and  $\Lambda_3$  such that  $\Lambda_1 \cup \Lambda_2 \cup \Lambda_3 = \mathcal{X}_{path}(\mathbf{S}, \mathbf{R})$ . The adversary slows down the messages in  $\Lambda_1$  and corrupts those in either  $\Lambda_2$  or  $\Lambda_3$ . Note that  $\mathbf{R}$  has no idea whether  $\Lambda_2$  is corrupted or  $\Lambda_3$  is corrupted and can not decide whether messages through  $\Lambda_2$  are correct or those through  $\Lambda_3$  are correct. Thus reliable transmission is not possible.

SUFFICIENT: See protocol in Fig. 6. Suppose  $\mathbf{S}$  transmits a message  $m$  to  $\mathbf{R}$ . Let  $\mathbf{R}$  receive  $m'$ . Assume that  $m' \neq m$ . This would require the adversary to corrupt the paths in  $\Lambda_B = (S_{recd} \setminus \Lambda'_D) = (\Phi(\mathbf{S}, \mathbf{R}) \setminus (\Lambda_D \cup \Lambda'_D))$ . Hence,  $\exists (\Lambda''_D, \Lambda''_E) \in \mathcal{A}_{path}^{[\Phi]}(\mathbf{S}, \mathbf{R})$  such that  $\Lambda''_D \supseteq \Lambda_B$ . This would imply that,  $\exists (\Lambda_D, \Lambda_E), (\Lambda'_D, \Lambda'_E), (\Lambda''_D, \Lambda''_E) \in \mathcal{A}_{path}^{[\Phi]}(\mathbf{S}, \mathbf{R})$  such that  $\Lambda_D \cup \Lambda'_D \cup \Lambda''_D = \Phi(\mathbf{S}, \mathbf{R})$ . This leads to a contradiction since  $\mathbf{S}$  and  $\mathbf{R}$  are  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected.  $\square$

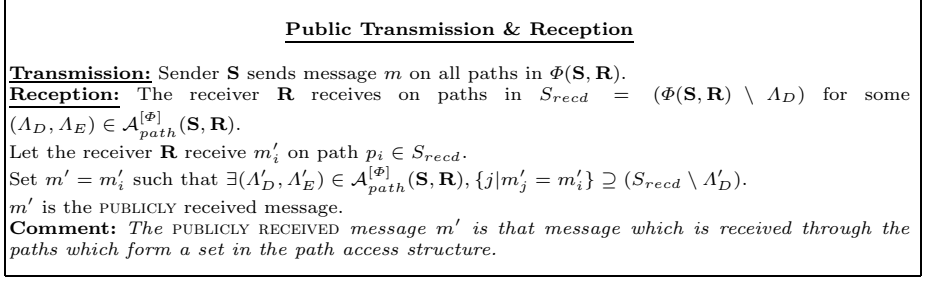


Fig. 6. Public Transmission and Reception.

## 7 Perfectly Secure Communication

### 7.1 Impossibility

**Theorem 7.** *Perfectly secure message transmission tolerating  $\mathcal{A}$  across an asynchronous network  $\mathcal{N}$  is possible only if the sender **S** and the receiver **R** are  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected as well as  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected. This is irrespective of whether  $\mathcal{N}$  has broadcast capabilities or not.*

PROOF: Assume for the sake of contradiction that there exists a scheme  $\xi$  for secure message transmission of  $m \in \mathcal{F}$  from **S** to **R** tolerating  $\mathcal{A}$  across  $\mathcal{N}$  not satisfying either  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnectivity or the  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnectivity condition. Also assume that each execution of  $\xi$  proceeds in phases, and that in the odd phase the sender **S** sends messages to the receiver **R** while in an even phase transmits to **S**.

*Case 1– Violation of  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnectivity.*

In this case, there exist three classes  $(\Lambda_{D_1}, \Lambda_{E_1}), (\Lambda_{D_2}, \Lambda_{E_2}), (\Lambda_{D_3}, \Lambda_{E_3}) \in \mathcal{A}_{path}^{\mathcal{X}}(\mathbf{S}, \mathbf{R})$  such that  $\Lambda_{D_1} \cup \Lambda_{D_2} \cup \Lambda_{D_3} = \mathcal{X}_{path}(\mathbf{S}, \mathbf{R})$ . Let  $m \neq m' \in \mathcal{F}$ . We construct two executions,  $\Psi$  and  $\Psi'$  of  $\xi$  that, for every  $k$ , are indistinguishable to **R** after  $k$  phases of communication. However, we construct the two executions in such a way that in  $\Psi$  the message being transmitted is  $m$ , while in  $\Psi'$  the message being transmitted is  $m'$  thus proving that these executions cannot terminate, violating the resiliency condition.

Assume that in phase  $2i + 1$  of the execution of  $\Psi$  (or  $\Psi'$ ), **S** sends  $\alpha_i$  (respectively  $\alpha'_i$ ) through the paths in  $\Lambda_{D_1}$ ,  $\beta_i$  (respectively  $\beta'_i$ ) through the paths in  $(\Lambda_{D_2} \setminus \Lambda_{D_1})$  and  $\gamma_i$  (respectively  $\gamma'_i$ ) through the paths in  $(\Lambda_{D_3} \setminus (\Lambda_{D_1} \cup \Lambda_{D_2}))$ . The adversary corrupts  $\Lambda_{D_3}$  (respectively  $\Lambda_{D_2}$ ) in the execution of  $\Psi$  (respectively  $\Psi'$ ) and delays the messages sent through paths in  $\Lambda_{D_1}$  so that **R** will not consider these messages. In each phase of the execution of  $\Psi$  (respectively  $\Psi'$ ), the adversary corrupts the message  $\gamma_i$  (respectively  $\beta_i$ ) to  $\gamma'_i$  (respectively  $\beta'_i$ ). At the end of the  $i^{th}$  phase, **R** receives  $\beta_i, \gamma'_i$  on paths in  $(\Lambda_{D_2} \cup \Lambda_{D_3}) \setminus \Lambda_{D_1}$  in both the executions. Clearly, the receiver cannot distinguish between the two executions, violating the *resiliency* requirement. Note that the existence of a broadcast channel does not help.



*Case 2– Violation of  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnectivity:*

This means that there exist two classes  $(\Lambda_{D_1}, \Lambda_{E_1}), (\Lambda_{D_2}, \Lambda_{E_2}) \in \mathcal{A}_{path}^{\mathcal{X}}(\mathbf{S}, \mathbf{R})$  such that  $\Lambda_{D_1} \cup \Lambda_{E_1} \cup \Lambda_{D_2} = \mathcal{X}_{path}(\mathbf{S}, \mathbf{R})$ . Hence, by choosing the class  $(\Lambda_{D_1}, \Lambda_{E_1})$  from  $\mathcal{A}$  and delaying the messages routed through paths in  $\Lambda_{D_2}$  (so that  $\mathbf{R}$  doesn't consider these messages), the adversary gains as much knowledge of the message as does  $\mathbf{R}$  violating the *secrecy* requirement.

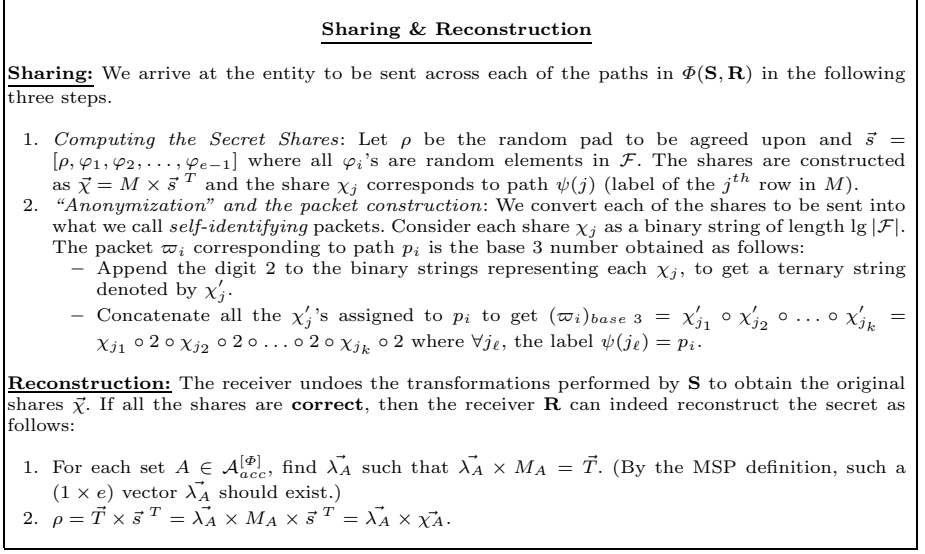
The fact that the presence of a broadcast primitive leaves the condition unaffected follows from Theorem 6 and the protocol (see Fig. 6) for simulating broadcast.  $\square$

## 7.2 Possibility

We show that the protocols in line with [5, 12], when modified to the asynchronous and generalized mixed adversary setting, works correctly over any network that is  $\mathcal{A}^{(2,1)}(\mathbf{S}, \mathbf{R})$ -subconnected as well as  $\mathcal{A}^{(3,0)}(\mathbf{S}, \mathbf{R})$ -subconnected. We begin by describing a method for sharing and reconstructing a secret, which we will denote as algorithm  $\mathcal{T}$  (see Fig. 7). It is known that a polynomial sized MSP<sup>6</sup> [11]  $\mathcal{M} = (\mathcal{F}, M_{d \times e}, \psi)$  can be constructed (preferably of size as small as possible), corresponding to the adversary structure  $\mathcal{A}_{path}^{[\mathcal{X}]}(\mathbf{S}, \mathbf{R})$  with the range of  $\psi$  being  $\Phi(\mathbf{S}, \mathbf{R})$  and the target vector  $\vec{T} = [1, 0, \dots, 0]$ . With these inputs, we describe the *sharing* and *reconstruction* algorithm  $\mathcal{T}$  in Fig. 7.

Our transmission protocol (see Fig. 8) runs in iterations. In each iteration the sender  $\mathbf{S}$  attempts the transmission of a random pad  $\rho$  by sending share  $\varpi_i$  (constructed using  $\mathcal{T}$ ) along each path  $p_i$ . Since,  $\mathbf{S}$  is not assured of hearing on all the paths,  $\mathbf{R}$  waits for messages on a subset of paths  $S_{recd}$  and attempts to reconstruct the random pad  $\rho$  (using the reconstruction algorithm of  $\mathcal{T}$ ) from the shares received. If  $\mathbf{R}$  is not able to conclusively reconstruct a unique pad,  $\mathbf{R}$  PUBLICLY SENDS all the received messages to  $\mathbf{S}$ . From these shares,  $\mathbf{S}$  constructs the set of faulty paths  $F$ . First  $\mathbf{S}$  constructs the set  $S_{recd}$ , the paths on which  $\mathbf{R}$  actually received messages. Among these paths,  $\mathbf{S}$  marks a path to be faulty (and adds to the set  $F$ ) if  $\mathbf{R}$  had received a wrong message. Note that at least one path is recognized as faulty, since otherwise the transmission of  $\rho$  would have been successful, terminating the pad-agreement phase of the protocol. Now  $\mathbf{S}$  and  $\mathbf{R}$  prune the adversary structure (as in the synchronous case) and restart the protocol for a different pad  $\rho'$ . When the transmission of a pad  $\rho$  is successful,

<sup>6</sup> Every linear secret sharing scheme can be represented as a Monotone Span Program defined as the triple  $(\mathcal{F}, M, \mathfrak{S})$  where  $\mathcal{F}$  represents a finite field,  $M$  is a  $d \times e$  matrix with entries in  $\mathcal{F}$ , and  $\mathfrak{S} : \{1 \dots d\} \rightarrow \{P_1 \dots P_n\}$  is a function. Each row of the matrix  $M$  is labeled by players in the sense that  $\mathfrak{S}(k)$  assigns the label  $\mathfrak{S}(k)$  to the  $k$ -th row of  $M$ ,  $1 \leq k \leq d$ . For  $A \subset \{P_1 \dots P_n\}$ ,  $M_A$  denotes the matrix that consists of all rows in  $M$  labeled by players in  $A$ . Let  $\vec{T} \in \mathcal{F}^e$  be the *target vector*. A MSP is said to accept (or reject) a structure  $\mathcal{Z}$  if  $\forall Z \in \mathcal{Z}$ , there exists (does not exist, respectively) a linear combination of the rows of  $M_Z$  which equals  $\vec{T}$ . An MSP is said to correspond to an adversary structure  $\mathcal{A}_{adv}$  if it rejects *exactly*  $\mathcal{A}_{adv}$ . By the *size* of an MSP, we mean the number of rows in  $M$ .



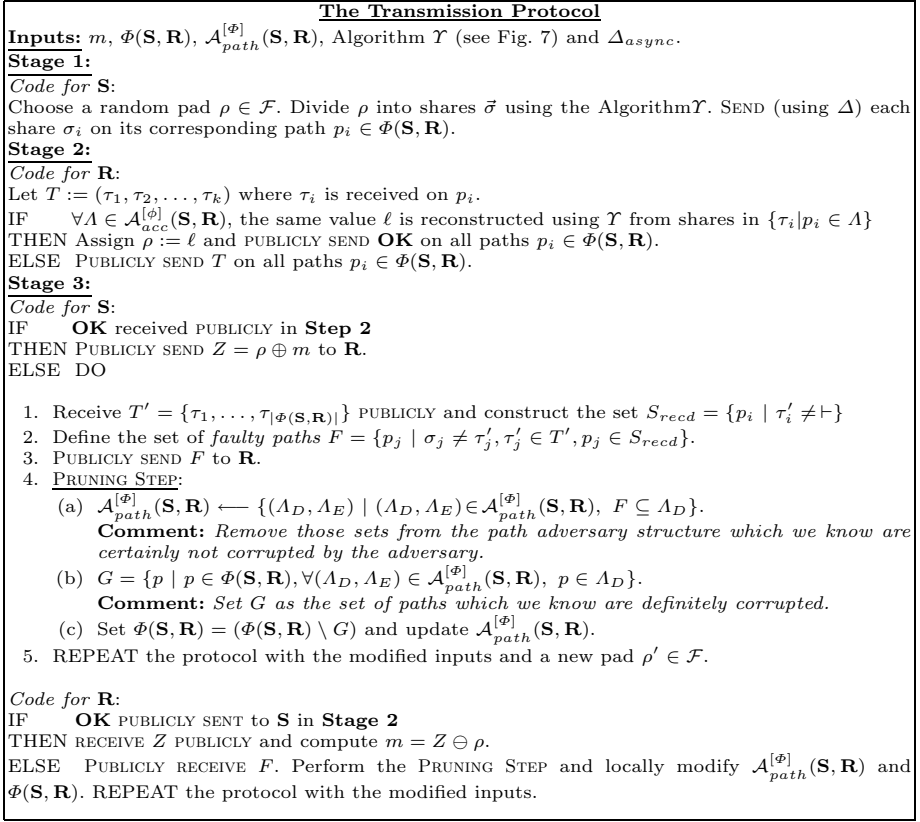
**Fig. 7.** Algorithm for Sharing and Reconstruction.

$\mathbf{R}$  PUBLICLY SENDS **OK** upon which  $\mathbf{S}$  sends  $Z = m \oplus \rho$  PUBLICLY. The receiver can get back  $m$  by  $m = Z \ominus \rho$ .

**Theorem 8.** *The transmission protocol given in Fig. 8 has the following properties: (1) The protocol provably terminates and runs in time polynomial in  $(n + |\mathcal{A}_{adv}|)$ . (2) The protocol satisfies the security requirements. (3) The overall message complexity of the protocol is polynomial in  $(n + |\mathcal{A}_{adv}|)$ . The proof follows from the Lemma 1, Theorem 1, Lemma 2 and Lemma 3.*

**Lemma 1 (Termination).** *The transmission protocol will terminate in at most  $|\mathcal{A}|$  iterations.*

PROOF: We show that if an iteration did not successfully transmit the random pad  $\rho$ , at least one faulty path will be detected. In each iteration, every path can be classified as an *OK* path ( $\mathbf{R}$  receives one message), or a *talkative* path ( $\mathbf{R}$  receives more than one message), or a *silent* path. The transmission of  $\rho$  will be successful if all the messages received on the *OK* paths were correct and there are no *talkative* paths. (We know that *silent* paths form a disruptive set in one of the classes in the path adversary structure.) If there is even one *talkative* path  $p_i$ , it would be marked with  $\tau_i = \perp$  and hence be recognized by  $\mathbf{S}$  as faulty. If any one of the messages on the *OK* paths are wrong, this path will be recognized as faulty since  $\mathbf{S}$  reliably receives all the messages received by  $\mathbf{R}$  (due to PUBLIC TRANSMISSION). Therefore, in each unsuccessful iteration, at least one faulty path is detected and thereby eliminated. Because of PRUNING STEP((b)&(c)) (see Fig. 8) the faulty path cannot occur in all the sets in  $\mathcal{A}_{path}^{[\Phi]}(\mathbf{S}, \mathbf{R})$ . This



**Fig. 8.** Perfectly Secure Transmission Protocol over Asynchronous Networks.

would result in the elimination of at least one set from the path-adversary in each iteration, because of PRUNING STEP(a). Hence, the algorithm will terminate in at most  $|\mathcal{A}_{path}^{[\Phi]}(\mathbf{S}, \mathbf{R})| = |\mathcal{A}|$  iterations.  $\square$

**Lemma 2 (Security).** *The protocol satisfies the resiliency and secrecy conditions for perfectly secure message transmission.*

**PROOF OF RESILIENCE:** The proof of resilience is similar to the one for the synchronous case. All that we need to prove is that whenever  $\mathbf{R}$  is able to successfully reconstruct a value of  $\rho'$ , then  $\rho' = \rho$ , i.e.,  $\mathbf{R}$  always reconstructs the correct value. We know that the path adversary structure satisfies  $\mathcal{Q}^{(3,0)} \cap \mathcal{Q}^{(2,1)}$  [7]. Exploiting the asynchrony of the network, the adversary can schedule the messages on the honest paths in some  $\Lambda_D$ , where  $(\Lambda_D, \Lambda_E) \in \mathcal{A}_{path}^{[\Phi]}(\mathbf{S}, \mathbf{R})$  and corrupt messages in some other paths in  $\Lambda'_D$ , where  $(\Lambda'_D, \Lambda'_E) \in \mathcal{A}_{path}^{[\Phi]}(\mathbf{S}, \mathbf{R})$ .

<sup>7</sup> The notation  $\mathcal{Q}^{(k,\ell)}$  has the same meaning as defined in [7].

By the definition of the corresponding access structure, there exists an access set  $\Lambda_A = \mathcal{P} \setminus (\Lambda_D \cup \Lambda'_D)$ . Clearly in our case, the messages  $\mathbf{R}$  receives through these paths are correct, i.e., they are the actual shares that  $\mathbf{S}$  sent. Hence, secret reconstructed using the MSP and this access set will be  $\rho$ , the pad that  $\mathbf{S}$  intended to send.  $\mathbf{R}$  reconstructs the secret correctly if and only if the secrets reconstructed using all the sets in the access structure are the same. Hence, if  $\mathbf{R}$  successfully reconstructs the secret, surely the reconstructed pad will be  $\rho$ .

**PROOF OF SECRECY:** Let the secret message to be transmitted be  $m \in \mathcal{F}$ . We first observe that in each of attempted transmissions of a random pad  $\rho$ , the adversary cannot “access” the secret (by definition of a MSP). Moreover, each of the  $\rho$  used in an iteration is independent of all the previous pads and the message  $m$ . Let  $r \in \mathcal{F}$  be a random field element. We claim that for any VIEW  $\mathcal{V}$  of  $\mathcal{A}$ ,  $\mathcal{V}$  occurs with the same probability in a transmission of  $m$  as in a transmission of  $m' = m \oplus r$ . Consider the case when the transmission of the pad  $\rho$  is successful. For the transmission of  $\rho$ ,  $\vec{s} = [\rho, \varphi_1, \dots, \varphi_{e-1}]$  while during the transmission of  $\rho'$ ,  $\vec{s}' = [\rho', \varphi'_1, \dots, \varphi'_{e-1}]$ .  $\mathbf{S}$  sends  $Z = m \oplus \rho = m \oplus (\vec{\lambda} \times M \times \vec{s}^T) = (m \oplus r) \oplus (r \oplus (\vec{\lambda} \times M \times \vec{s}^T)) = (m') \oplus (\vec{\lambda} \times M \times \vec{s}'^T) = m' \oplus \rho'$ .  $\square$

**Lemma 3 (Communication Complexity).** *Each iteration of the protocol communicates polynomial in  $(n + |\mathcal{A}|)$  bits.*

**PROOF:** From Theorem 1, it is clear that total number of paths used in the transmission protocol is polynomial in the size of  $\mathcal{A}$ . In **Stage 1** of every iteration, we have,  $\mathbf{S}$  sends  $O(|\Phi(\mathbf{S}, \mathbf{R})|)$  field elements to  $\mathbf{R}$ . In **Stage 2**,  $\mathbf{R}$  replies with  $O((|\Phi(\mathbf{S}, \mathbf{R})|)^2)$  field elements. Since size of  $\Phi(\mathbf{S}, \mathbf{R})$  is polynomial in the size of  $\mathcal{A}$ , the communication complexity of the protocol is polynomial in the size of the input.  $\square$

## 8 Conclusion

Network synchrony is a very difficult primitive to achieve in real-life, and more so in the presence of Byzantine faults in the system. This work initiates and completely characterizes the minimum connectivity requirements for secure communication over completely asynchronous networks (see Table 1). Furthermore, the choice of generalized mixed adversaries has meant that the necessary and sufficient conditions for secure communication over incomplete asynchronous networks for a variety of adversarial settings is studied in an unified manner and expressed in one-shot. The study of information-theoretically secure communication is far from closed. We have not considered the third kind of fundamental faulty behaviour, viz. fail-stop faults. Another open thread yet to be explored is to suitably adapt the protocols to (more practical) settings with lower amounts of synchrony though not completely asynchronous. Such networks are called *partially synchronous* networks. Yet another interesting setting is one in which the players possess only a partial knowledge of the topology of the network. More

**Table 1.** Necessary and sufficient connectivity requirements for the possibility of secure communication between any two honest players over arbitrary asynchronous networks.

	Perfect security	Unconditional security	Unconditional with Broadcast
Threshold Adversary	$\max(t_a, t_p) + 2t_a + 1$ [13]	$\max(t_a, t_p) + 2t_a + 1$	$2t_a + t_p + 1$
Generalized Adversary	$(\mathcal{A}^{(3,0)} \ \& \ \mathcal{A}^{(2,1)})$	$(\mathcal{A}^{(3,0)} \ \& \ \mathcal{A}^{(2,1)})$	$(\mathcal{A}^{(2,1)})$

realistic adversary models are worth exploring and will have repercussions not only to the secure communication problem but also in the field of secure multiparty computation. Extant adversary models characterize a deviant player as either an *honest* player or a *dishonest* player. However, in real-life players being “fairly honest” and “slightly dishonest” makes sense. Viewing the honesty of the players with this fuzzy outlook is worth exploring. Furthermore, among the efficiency considerations, it would be worth investigating the direct-sum question with respect to the communication as well as randomness complexities. Moreover, our protocols (for the perfect security case) are based on perfect linear secret schemes. This work does not investigate the deployment of non-linear secret sharing schemes that may prove to be more efficient (see [11]).

## References

1. A. Beimel and Y. Ishai. On the power of nonlinear secret sharing. In *16th Annual IEEE Structure in Complexity Theory*, pages 188–202, 2001.
2. M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computations. In *25th ACM STOC*, pages 52–61, 1993.
3. M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous secure computation with optimal resilience. In *13th ACM PODC*, pages 183–192, 1994.
4. C. Blundo and D. R. Stinson. Anonymous secret sharing schemes. *Discrete Applied Mathematics*, 77:13–28, 1997.
5. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *JACM*, volume 40, number 1, pages 17–47, 1993.
6. M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, volume 32, number 2, pages 374–382, 1985.
7. M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multiparty computation. In *ASIACRYPT’99*, volume 1716 of LNCS. Springer-Verlag, 1999.
8. M. Franklin and R. N. Wright. Secure communication in minimal connectivity models. *Journal of Cryptology*, 13(1):9–30, 2000.
9. M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, April 2000. Preliminary version appeared in *16th ACM PODC*, pages 25–34, August 1997.
10. M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *IEEE Globecom 87*, pages 99–102. IEEE, 1987.

11. M. Karchmer and A. Wigderson. On span programs. In *8th IEEE Structure in Complexity Theory*, pages 102–111, 1993.
12. M.V.N. Ashwin Kumar, P.R. Goundan, K. Srinathan, and C.P. Rangan. On perfectly secure communication over arbitrary networks. In *21st ACM PODC*, 2002.
13. H. Sayeed and H. Abu-Amara. Perfectly secure message transmission in asynchronous networks. In *7th IEEE Symposium on Parallel and Distributed Processing*, 1995.

# Amplified Boomerang Attack against Reduced-Round SHACAL

Jongsung Kim, Dukjae Moon, Wonil Lee,  
Seokhie Hong, Sangjin Lee, and Seokwon Jung

Center for Information Security Technologies (CIST), Korea University,  
Anam Dong, Sungbuk Gu, Seoul, Korea,  
{joshep,djmoon,nice,hsh,sangjin,jsw}@cist.korea.ac.kr

**Abstract.** SHACAL is a 160-bit block cipher based on the hash standard SHA-1, as a submission to NESSIE. SHACAL uses the XOR, modular addition operation and the functions of bit-by-bit manner. These operations and functions make the differential cryptanalysis difficult, i.e., it is hard to find a long differential characteristic with high probability. But, we can find short differential characteristics with high probabilities. Using this fact, we discuss the security of SHACAL against an amplified boomerang attack. We find a 36-step boomerang-distinguisher and present attacks on reduced-round SHACAL with various key sizes. We can attack 39-step SHACAL with 256-bit key, and 47-step SHACAL with 512-bit key. In addition, we present differential attacks of reduced-round SHACAL with various key sizes.

**Keyword:** SHACAL, Amplified boomerang attack, Boomerang-distinguisher

## 1 Introduction

SHACAL [3] is a 4-round block cipher (each line consists of 20 steps.) designed by H. Handschuh and D. Naccache and is one of the accepted NESSIE submissions. SHACAL was designed by using the hash standard SHA-1 in encryption mode for the first time in 2000. Also, H. Handschuh and D. Naccache introduced a modification [4] of SHACAL in its two versions SHACAL-1 and SHACAL-2 in 2001. In its basic version, SHACAL-1 is a 160-bit block cipher based on SHA-1 and in its extended version, SHACAL-2 is a 256-bit block cipher based on SHA-2. In this paper, we only attack reduced-round SHACAL-1. We will just call SHACAL-1 as SHACAL.

The main cryptanalytic results obtained on SHACAL so far are the analysis of the differential and linear attacks by the algorithm designers [3], and statistical evaluation by J. Nakahara Jr [7]. In [3], the algorithm designers proposed 10-step linear approximations with bias  $2^{-6}$  in rounds 1, 2 and 4 respectively, and a 10-step linear approximation with bias  $2^{-5}$  in round 3. Also, they proposed a 10-step differential characteristic with probability  $2^{-13}$  in rounds 1 and 3, and a 10-step differential characteristic with probability  $2^{-26}$  in rounds 2 and 4. Using these 10-step linear approximations and differential characteristics, they concluded that a

**Table 1.** Our result of attacks on reduced-round SHACAL

Master Key	Steps	Methods	Data	Time
128-bit	28	Amp.Boo.	$2^{127.5}$	$2^{127.2}$
128-bit	30	DC	$2^{110}$	$2^{75.1}$
160-bit	37	Amp.Boo.	$2^{158.8}$	$2^{87.8}$
160-bit	32	DC	$2^{141}$	$2^{105}$
256-bit	39	Amp.Boo.	$2^{158.5}$	$2^{250.8}$
256-bit	34	DC	$2^{141}$	$2^{234}$
512-bit	47	Amp.Boo.	$2^{158.5}$	$2^{508.4}$
512-bit	41	DC	$2^{141}$	$2^{491}$

linear attack with less than  $2^{80}$  known plaintexts is not applicable to full-round SHACAL, and that a differential attack with less than  $2^{116}$  chosen plaintexts is not applicable to full-round SHACAL.

In this paper, we propose a 10-step differential characteristic with probability  $2^{-12}$  in rounds 2 and 4. This characteristic has much higher probability than one proposed by the algorithm designers. Using this characteristic, we describe a 36-step boomerang-distinguisher. We use this boomerang-distinguisher to devise amplified boomerang attacks on reduced-round SHACAL with various key sizes. Moreover, we present a differential attack and compare the results of an amplified boomerang attack with those of a differential attack. Table 1 summarizes attacks on reduced-round SHACAL with respect to master key sizes. Amplified Boomerang attack is denoted by Amp.Boo. in Table 1, and a time complexity of  $n$  means that the time of an attack corresponds to performing  $n$  encryptions of the underlying cipher.

2 Preliminaries

2.1 Description of SHACAL

SHA is a hash function which was introduced by the American National Institute for Standards and Technology in 1993, and is known as SHA-0. In 1995, a minor change to SHA-0 was made, this variant known as SHA-1. The standard now includes only SHA-1. SHACAL is a 160-bit block cipher based on the hash standard SHA-1. Description of SHACAL [3] is as follows.

Notation:

- $+$ : Addition modulo  $2^{32}$  of 32-bit words.
- $ROT_i(X)$ : Rotate 32-bit word  $X$  to the left by  $i$ -bit positions.
- $\oplus$ : Bitwise exclusive-or.
- $\&$ : Bitwise and.
- $|$ : Bitwise or.

The procedure to encrypt a message is as follows.



1. Insert the 160-bit message  $X(= X_1||X_2||X_3||X_4||X_5)$  where each  $X_i$  is a 32-bit word in the 32-bit words,  $A_0, B_0, C_0, D_0, E_0$ , by

$$A_0 = X_1, B_0 = X_2, C_0 = X_3, D_0 = X_4, E_0 = X_5.$$

2. Encrypt the 32-bit words,  $A_0, B_0, C_0, D_0, E_0$  in a total of 80 steps. So, we have a ciphertext,  $A_{80}, B_{80}, C_{80}, D_{80}, E_{80}$ . Encryption process of the  $i^{th}$  step is as follows.

$$A_i = K_i + ROT_5(A_{i-1}) + f_i(B_{i-1}, C_{i-1}, D_{i-1}) + E_{i-1} + y_i$$

$$B_i = A_{i-1}$$

$$C_i = ROT_{30}(B_{i-1})$$

$$D_i = C_{i-1}$$

$$E_i = D_{i-1}$$

for  $i = 1, \dots, 80$ , where

$$f_i(B, C, D) = (B \& C) | (-B \& D), \quad (1 \leq i \leq 20)$$

$$f_i(B, C, D) = B \oplus C \oplus D, \quad (21 \leq i \leq 40, 61 \leq i \leq 80)$$

$$f_i(B, C, D) = (B \& C) | (B \& D) | (C \& D), \quad (41 \leq i \leq 60)$$

We call each  $f_i$  as  $f_{if}$  ( $1 \leq i \leq 20$ ),  $f_{xor}$  ( $21 \leq i \leq 40, 61 \leq i \leq 80$ ), and  $f_{maj}$  ( $41 \leq i \leq 60$ ), respectively. Each  $K_i$  is a 32-bit subkey of the  $i^{th}$  step. Each constant  $y_i$  is defined as

$$y_i = 5a827999_x, \quad (1 \leq i \leq 20)$$

$$y_i = 6ed9eba1_x, \quad (21 \leq i \leq 40)$$

$$y_i = 8f1bbcdc_x, \quad (41 \leq i \leq 60)$$

$$y_i = ca62c1d6_x, \quad (61 \leq i \leq 80)$$

The key scheduling of SHACAL takes a maximum 512-bit key and shorter keys may be used by padding the key with zeros to a 512-bit string. However, SHACAL is not intended to be used with a key shorter than 128 bits. Let the 512-bit key string be denoted  $K = [K_1||K_2||\dots||K_{16}]$ , where each  $K_i$  is a 32-bit word. The key expansion of 512 bits  $K$  to 2560 bits is defined by

$$K_i = ROT_1(K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}), \quad (17 \leq i \leq 80)$$

## 2.2 Amplified Boomerang Attack

The amplified boomerang attack [6] is a chosen plaintext attack, while the boomerang attack [8] is an adaptive chosen plaintext and ciphertext attack. The main idea of the amplified boomerang attack is to use two short differential characteristics with high probabilities instead of a long characteristic with low probability.

Let a block cipher  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be composed of a cascade  $E = E_1 \circ E_0$ . We assume that for  $E_0$  there exists a differential characteristic  $\alpha \rightarrow \beta$  with probability  $p$ , and for  $E_1$  there exists a differential characteristic  $\gamma \rightarrow \delta$  with probability  $q$ , where  $pq \gg 2^{-n/2}$ .

The amplified boomerang attack is based on building quartets of plaintexts  $(X_1, X_2, X_3, X_4)$  which satisfy several differential conditions. Assume that  $X_1 \oplus X_2 = \alpha$  and  $X_3 \oplus X_4 = \alpha$ . We denote by  $X'_1, X'_2, X'_3, X'_4$  the encrypted values of  $X_1, X_2, X_3, X_4$  under  $E_0$  respectively, and by  $X''_1, X''_2, X''_3, X''_4$  the encrypted values of  $X'_1, X'_2, X'_3, X'_4$  under  $E_1$  respectively. We are interested in the cases where  $X'_1 \oplus X'_2 = X'_3 \oplus X'_4 = \beta$  and  $X'_1 \oplus X'_3 = \gamma$  (or  $X'_1 \oplus X'_4 = \gamma$ ), as in these cases  $X'_2 \oplus X'_4 = (X'_1 \oplus \beta) \oplus (X'_3 \oplus \beta) = \gamma$  (or  $X'_2 \oplus X'_3 = \gamma$ ) as well. If the output difference of  $E_1$  becomes  $\delta$  when the input difference is  $\gamma$ , i.e.  $X''_1 \oplus X''_3 = X''_2 \oplus X''_4 = \delta$  (or  $X''_1 \oplus X''_4 = X''_2 \oplus X''_3 = \delta$ ), a quartet satisfying all these differential conditions is called a right quartet. An description of such a quartet is shown in Fig. 1.

If we have  $m$  pairs with difference  $\alpha$ , we can calculate the fraction of the right quartets among all the quartets generated by  $m$  pairs. First, we have about  $mp$  pairs satisfying a differential characteristic  $\alpha \rightarrow \beta$  for  $E_0$ . The  $mp$  pairs generate about  $(mp)^2/2$  quartets consisting of two such pairs. Assuming that the intermediate encryption values distribute uniformly over all possible values, we get  $X'_1 \oplus X'_3 = \gamma$  or  $X'_1 \oplus X'_4 = \gamma$  with probability  $2^{-n+1}$ . Second, for the  $((mp)^2/2) \cdot 2^{-n+1}$  quartets satisfying above differential conditions, we can get

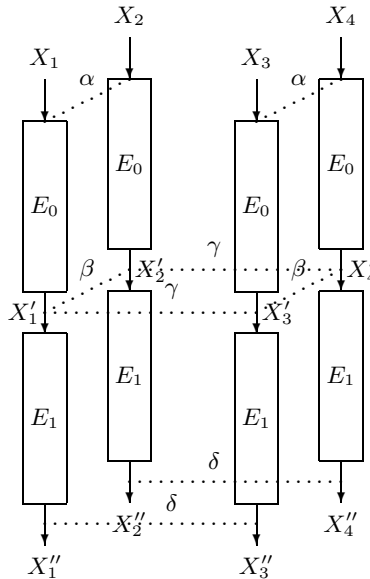


Fig. 1. Boomerang-Distinguisher

right quartets with probability  $q^2$  by the characteristic for  $E_1$ . Therefore, the expected number of right quartets is about  $m^2 \cdot 2^{-n} \cdot (pq)^2$ .

For a random permutation, the expected number of right quartets is about  $m^2 \cdot 2^{-2n} (= (m^2/2) \cdot 2^{-2n+1})$ . Therefore, if  $pq > 2^{-n/2}$  and  $m$  is sufficiently large, we can have a boomerang-distinguisher which distinguishes between  $E$  and a random cipher.

### 3 Amplified Boomerang Attacks on SHACAL

We describe the differential properties of two operations and three step functions used in SHACAL. We find a 36-step boomerang-distinguisher of SHACAL using these properties and attack reduced round SHACAL.

#### 3.1 Differential Properties for SHACAL

We present two differential properties used in generating a differential characteristic of SHACAL. What generates a differential probability on SHACAL is first, the use of both XOR and modular additions, and second, the functions  $f_{if}, f_{xor}, f_{maj}$ .

First, we consider the relation between XOR differences and modular addition. Let  $X, Y$  and  $X^*, Y^*$  be 32-bit words. We assume  $Z = X + Y$  and  $Z^* = X^* + Y^*$ . If the words  $X$  and  $Y$  only differ in the position of bit  $i$  ( $0 \leq i \leq 31$ ), we denote by  $X \oplus Y = e_i$  where the most significant bit (left) is a bit of position 31. Then, we have the following four relations [5] between XOR differences and modular addition. In the relations 3 and 4, the  $j$  indicates  $0 \leq j \leq 30$ .

1. If  $X \oplus X^* = e_{31}$  and  $Y = Y^*$ , then it holds  $Z \oplus Z^* = e_{31}$  with probability 1.
2. If  $X \oplus X^* = e_{31}$  and  $Y \oplus Y^* = e_{31}$ , then it holds  $Z = Z^*$  with probability 1.
3. If  $X \oplus X^* = e_j$  and  $Y = Y^*$ , then it holds  $Z \oplus Z^* = e_j$  with probability  $1/2$ .
4. If  $X \oplus X^* = e_j$  and  $Y \oplus Y^* = e_j$ , then it holds  $Z = Z^*$  with probability  $1/2$ .

Second, we consider differential probabilities for the functions  $f_{if}, f_{xor}, f_{maj}$ . These functions operate in the bit-by-bit manner. Thus, we can regard each  $f_i$  as a boolean function assigning from a 3-bit input to a 1-bit output. Table 2 [5] shows distribution of XOR differences through all three functions. The notation of the table is as follows. The first three columns represent the eight possible differences in the one-bit inputs,  $x, y, z$ . The next three columns indicate the differences in the outputs of each of the three functions. In the last three columns, a '0'('1') means that the difference will always be zero(one), and a '0/1' means that in half of the cases, the difference will be zero and in the other half of the cases, the difference will be one.

#### 3.2 The 36-Step Boomerang-Distinguisher

Using the differential properties shown in the previous subsection, we describe two differential characteristics which make a boomerang-distinguisher for SHACAL. That is, the first differential characteristic is  $\alpha \rightarrow \beta$  with probability

**Table 2.** The XOR differential distribution table of the  $f$ -functions

$x$	$y$	$z$	$f_{xor}$	$f_{if}$	$f_{maj}$
0	0	0	0	0	0
0	0	1	1	0/1	0/1
0	1	0	1	0/1	0/1
1	0	0	1	0/1	0/1
0	1	1	0	1	0/1
1	0	1	0	0/1	0/1
1	1	0	0	0/1	0/1
1	1	1	1	0/1	1

$p$  ( $= 2^{-45}$ ) from steps 1 to 21, where the differences  $\alpha = (0, e_{22}, e_{15}, e_{10}, e_5)$  and  $\beta = (e_{2,7,14,24,29}, e_{19}, e_{12}, e_7, e_2)$  where  $e_{i_1, \dots, i_k}$  indicates  $e_{i_1} \oplus \dots \oplus e_{i_k}$ . The second differential characteristic is  $\gamma \rightarrow \delta$  with probability  $q$  ( $= 2^{-31}$ ) from steps 22 to 36, where the differences  $\gamma = (e_{1,5,8}, e_{1,3,5}, e_{3,13}, e_{1,5,13,31}, e_{6,10,13,31})$  and  $\delta = (e_{9,19,29,31}, e_{14,29}, e_{7,29}, e_2, e_{29})$ . Table 3 shows the first differential characteristic composed of 21 steps. In Table 3 the first row indicates an input difference of the 1<sup>st</sup> step, and the second column of the  $i^{th}$  step indicates an output difference of the  $i^{th}$  step, and the third column of the  $i^{th}$  step indicates the probability

**Table 3.** The first differential characteristic for SHACAL

Step	$\Delta A$	$\Delta B$	$\Delta C$	$\Delta D$	$\Delta E$	Prob
	0	$e_{22}$	$e_{15}$	$e_{10}$	$e_5$	
1	$e_5$	0	$e_{20}$	$e_{15}$	$e_{10}$	$2^{-4}$
2	0	$e_5$	0	$e_{20}$	$e_{15}$	$2^{-3}$
3	$e_{15}$	0	$e_3$	0	$e_{20}$	$2^{-3}$
4	0	$e_{15}$	0	$e_3$	0	$2^{-2}$
5	0	0	$e_{13}$	0	$e_3$	$2^{-2}$
6	$e_3$	0	0	$e_{13}$	0	$2^{-2}$
7	$e_8$	$e_3$	0	0	$e_{13}$	$2^{-2}$
8	0	$e_8$	$e_1$	0	0	$2^{-2}$
9	0	0	$e_6$	$e_1$	0	$2^{-2}$
10	0	0	0	$e_6$	$e_1$	$2^{-2}$
11	$e_1$	0	0	0	$e_6$	$2^{-2}$
12	0	$e_1$	0	0	0	$2^{-1}$
13	0	0	$e_{31}$	0	0	$2^{-1}$
14	0	0	0	$e_{31}$	0	$2^{-1}$
15	0	0	0	0	$e_{31}$	$2^{-1}$
16	$e_{31}$	0	0	0	0	1
17	$e_4$	$e_{31}$	0	0	0	$2^{-1}$
18	$e_9$	$e_4$	$e_{29}$	0	0	$2^{-2}$
19	$e_{14}$	$e_9$	$e_2$	$e_{29}$	0	$2^{-3}$
20	$e_{19}$	$e_{14}$	$e_7$	$e_2$	$e_{29}$	$2^{-4}$
21	$e_{2,7,14,24,29}$	$e_{19}$	$e_{12}$	$e_7$	$e_2$	$2^{-5}$

**Table 4.** The second differential characteristic for SHACAL

Step	$\Delta A$	$\Delta B$	$\Delta C$	$\Delta D$	$\Delta E$	Prob
	$e_{1,5,8}$	$e_{1,3,5}$	$e_{3,13}$	$e_{1,5,13,31}$	$e_{6,10,13,31}$	
22	0	$e_{1,5,8}$	$e_{1,3,31}$	$e_{3,13}$	$e_{1,5,13,31}$	$2^{-3}$
23	$e_{1,8}$	0	$e_{3,6,31}$	$e_{1,3,31}$	$e_{3,13}$	$2^{-4}$
24	$e_{1,3}$	$e_{1,8}$	0	$e_{3,6,31}$	$e_{1,3,31}$	$2^{-4}$
25	0	$e_{1,3}$	$e_{6,31}$	0	$e_{3,6,31}$	$2^{-4}$
26	$e_1$	0	$e_{1,31}$	$e_{6,31}$	0	$2^{-3}$
27	$e_1$	$e_1$	0	$e_{1,31}$	$e_{6,31}$	$2^{-2}$
28	0	$e_1$	$e_{31}$	0	$e_{1,31}$	$2^{-1}$
29	0	0	$e_{31}$	$e_{31}$	0	$2^{-1}$
30	0	0	0	$e_{31}$	$e_{31}$	1
31	0	0	0	0	$e_{31}$	1
32	$e_{31}$	0	0	0	0	1
33	$e_4$	$e_{31}$	0	0	0	$2^{-1}$
34	$e_{9,31}$	$e_4$	$e_{29}$	0	0	$2^{-1}$
35	$e_{14,29}$	$e_{9,31}$	$e_2$	$e_{29}$	0	$2^{-3}$
36	$e_{9,19,29,31}$	$e_{14,29}$	$e_{7,29}$	$e_2$	$e_{29}$	$2^{-4}$

with which an output difference of the  $(i - 1)^{th}$  step becomes an output difference of the  $i^{th}$  step. Note that the function  $f_{if}$  is used from steps 1 to 20, and the function  $f_{xor}$  is used at the  $21^{th}$  step. We can easily check probabilities in Table 3 using the differential properties on SHACAL. Thus, we have the first differential characteristic  $\alpha \rightarrow \beta$  with probability  $p (= 2^{-45})$  from steps 1 to 21 shown in Table 3.

Table 4 shows the second differential characteristic composed of 15 steps. Note that the function  $f_{xor}$  is used from steps 22 to 36. Similarly, we can have the second differential characteristic  $\gamma \rightarrow \delta$  with probability  $q (= 2^{-31})$  from steps 22 to 36 shown in Table 4.

Two differential characteristics above can be regarded as extended ones for 10-step differential characteristics with high probabilities respectively. That is, in the first differential characteristic, the good 10-step characteristic is  $(0, e_8, e_1, 0, 0) \rightarrow (e_9, e_4, e_{29}, 0, 0)$  with probability  $2^{-13}$  from steps 9 to 18, and in the second differential characteristic, the good 10-step characteristic is  $(0, e_{1,3}, e_{6,31}, 0, e_{3,6,31}) \rightarrow (e_{14,29}, e_{9,31}, e_2, e_{29}, 0)$  with probability  $2^{-12}$  from steps 26 to 35. Especially, the 10-step characteristic from steps 26 to 35 has much higher probability than one proposed by algorithm designers [3]. Also, if we extend the differential characteristics in Table 3, 4 to more steps, hamming weights in the differences of the five words become much bigger and the probabilities decrease rapidly. In the heuristic point of view, we conjecture that the 36-step boomerang-distinguisher using two differential characteristics in Table 3, 4 is one of the longest boomerang-distinguishers such that  $pq \gg 2^{-80}$  for SHACAL.

### 3.3 Attack Procedure

We present here amplified boomerang attacks on reduced-round SHACAL with various key sizes. We now present a method to use the 36-step boomerang-distinguisher to find subkey material.

Let  $S = E_f \circ E = E_f \circ E_1 \circ E_0$  be reduced-round SHACAL such that  $E_0$  indicates from steps 1 to 21, and  $E_1$  indicates from steps 22 to 36. We find the subkey material of  $E_f$  in  $S$ . The first differential characteristic  $\alpha \rightarrow \beta$  used in  $E_0$  has the probability  $p (= 2^{-45})$  and the second differential characteristic  $\gamma \rightarrow \delta$  used in  $E_1$  has the probability  $q (= 2^{-31})$ . The differences  $\alpha, \beta, \gamma$  and  $\delta$  are presented in the subsection 3.2. So, we have the 36-step boomerang-distinguisher with probability  $pq (= 2^{-76})$  from steps 1 to 36.

For  $m = 2^{157.5}$  pairs with the input difference  $\alpha$ , the expected number of right quartets is 8  $(= (2^{157.5})^2 \cdot 2^{-160} \cdot (2^{-76})^2)$ . From this fact, we can construct an algorithm to attack  $S$  with at least 160 bits key as follows.

**1. Choose  $m (= 2^{157.5})$  pairs with the input difference  $\alpha$ .**

The expected number of possible quartets from the pool of  $m$  pairs is about  $m^2 (= 2^{315})$ . We denote the plaintexts of a quartet by  $(P_1, P_2, P_3, P_4)$  where  $P_1 \oplus P_2 = P_3 \oplus P_4 = \alpha$  and the corresponding ciphertexts by  $(C_1, C_2, C_3, C_4)$ .

**2. Initialize the counter array with 0's.**

The number of the counter array is equal to the number of possible keys for  $E_f$ .

**3. Check the differences  $C_1 \oplus C_3 = C_2 \oplus C_4 = \delta'$  where  $\delta'$  is an element of the set composed of possible output differences for  $E_f$  with the input difference  $\delta$   $(= (e_{9,19,29,31}, e_{14,29}, e_{7,29}, e_2, e_{29}))$ .**

**4. For all the quartets which passed the last test, increase the counters by 1 which correspond to all subkeys  $K_f$  of  $E_f$  for which  $E_{f_{K_f}}^{-1}(C_1) \oplus E_{f_{K_f}}^{-1}(C_3) = E_{f_{K_f}}^{-1}(C_2) \oplus E_{f_{K_f}}^{-1}(C_4) = \delta$ .**

**5. Check all counters, and output the subkey whose counter is greater than or equal to 7.**

First, using this algorithm, we show that the reduced 39-step SHACAL with 256-bit key can be broken by an attack which is faster than an exhaustive search for a master key. Since  $E_f$  consists of the 37<sup>th</sup>, 38<sup>th</sup> and 39<sup>th</sup> steps, we can find the 96-bit subkey  $K_f$ .

In Step 1, we have  $2^{315}$  quartets derived from  $2^{157.5}$  pairs with the difference  $\alpha$ . For these quartets, we can filter out wrong quartets through Step 3. In Step 3, we take  $\delta'$  that belongs to the set  $\{(\cdot, \cdot, \cdot, e_{7,17,27,29}, e_{12,27}) \mid \cdot \text{ is an arbitrary difference} \}$  composed of possible output differences for  $E_f$  with the input difference  $\delta$ . So, we have  $2^{187}$  candidates for right quartets among  $2^{315}$  quartets, since a fraction of  $(2^{-64})^2$  of these quartets remain. In Step 4, we guess a 96-bit subkey  $K_f$  and decrypt ciphertexts of the remaining quartets for guessed key. If a decrypted quartet passes through Step 4, the counter of guessed key is increased by 1. So, the expected value of counter of right subkey is greater than 7, since the expected number of right quartets is about 8. But, for a wrong subkey, the expected value of counter is equal to 0 or 1, since the expected number of

quartets passed through Step 4 is  $2^{-5}(= 2^{187} \cdot (2^{-96})^2)$ . Thus, we can find the right key of  $E_f$  by the maximum likelihood method. The attack requires  $2^{158.5}$  chosen plaintexts and processing equivalent to about  $2^{158.5} \cdot 2^{96} \cdot \frac{3}{39} \simeq 2^{250.8}$  39-step SHACAL encryptions.

Also, using the algorithm above, we can attack on reduced-round SHACAL with at least 256-bit keys. We assume that for  $i = 0, 1, \dots, 8$ , the reduced  $(39+i)$ -step SHACAL uses the  $(256 + 32 \cdot i)$ -bit master key. Since  $E_f$  consists of  $(i + 3)$  steps, we can find the  $(32 \cdot (i + 3))$ -bit subkey  $K_f$  for the reduced  $(39 + i)$ -step SHACAL by the algorithm above. Particularly, in the algorithm for the reduced  $(39 + i)$ -step SHACAL ( $i \geq 2$ ), there does not exist the filtering process (Step 3) since we use the 36-step boomerang-distinguisher to attack. The attack for  $(39+i)$ -step SHACAL requires  $2^{158.5}$  chosen plaintexts and processing equivalent to about  $2^{158.5} \cdot 2^{32 \cdot (i+3)} \cdot \frac{i+3}{39+i} (\leq 2^{252.4+32 \cdot i})$   $(39 + i)$ -step SHACAL encryptions where  $i = 0, 1, \dots, 8$ . Thus we can attack the reduced 47-step SHACAL with 512-bit key. Furthermore, we can attack on reduced-round SHACAL with less than 256-bit key except 128-bit key. In these cases, since the key sizes are small, the expected number of quartets passed through Step 3 (filtering process) should be less than  $2^{156.5}$  to attack reduced-round SHACAL faster than the exhaustive search. Thus, we can attack the reduced 37-step SHACAL with 160-bit key and the reduced 38-step SHACAL with 192- or 224-bit master key. The attack for 37-step SHACAL requires  $2^{158.5}$  chosen plaintexts and processing equivalent to about  $2^2 \cdot 2^{315} \cdot 2^{-256} \cdot 2^{32} \cdot \frac{1}{37} \simeq 2^{87.8}$  37-step SHACAL encryptions, and the attack for 38-step SHACAL requires  $2^{158.5}$  chosen plaintexts and processing equivalent to about  $2^2 \cdot 2^{315} \cdot 2^{-192} \cdot 2^{64} \cdot \frac{2}{38} \simeq 2^{184.8}$  38-step SHACAL encryptions.

In the case of 128-bit key, we cannot use the above 36-step boomerang-distinguisher since the number of required plaintexts should be less than  $2^{128}$ . So, we must find a new boomerang-distinguisher with probability  $pq$  which is higher than  $2^{-45.5} (= \{2^3 \cdot (2^{-127})^2 \cdot 2^{160}\}^{1/2})$ . We can find a 26-step boomerang-distinguisher with probability  $2^{-45}$  from steps 1 to 26. We can attack on 28-step SHACAL. Since differential attack which is described in the next section is applied to SHACAL more effective than amplified boomerang attack, we omit the detailed explanation. See table 1 for the result of an attack on SHACAL with 128-bit key.

## 4 Differential Attacks on SHACAL

In this section, we present differential attacks on reduced-round SHACAL. First of all, we describe two differential characteristics which are expanded from the 21-step differential characteristic shown in Table 3. One is the 28-step differential characteristic  $\alpha \rightarrow \beta$  1 with probability  $2^{-107}$  from steps 1 to 28, the other is the 30-step differential characteristic  $\alpha \rightarrow \beta'$  2 with probability  $2^{-138}$  from steps 1

<sup>1</sup>  $\beta' = (e_{0,2,5,15,19,20,22,25,27}, e_{3,7,12,14,25,29,30}, e_{8,10,25}, e_{5,8,12,20,27}, e_{0,3,17,25,30})$

<sup>2</sup>  $\beta'' = (e_{0,1,3,6,15,23,27,28,29}, e_{0,1,14,17,24,29,30}, e_{0,3,13,17,18,20,23,25,30}, e_{1,5,10,12,23,27,28}, e_{8,10,25})$

to 30. We can easily check probabilities of these differential characteristics using the differential properties on SHACAL.

Using the 28-step differential characteristic, we show that the reduced 30-step SHACAL with 128-bit key can be broken by a differential attack which is faster than an exhaustive search for a master key. That is, we can find the 64-bit subkey of the  $29^{th}$  and  $30^{th}$  steps. Note that these steps are denoted by  $E_f$ . Attack procedure is as follows. First, we ask for  $2^{109}$  pairs with the input difference  $\alpha$ . Second, we check whether the output differences of these pairs are equal to  $(?, ?, e_{0,3,13,17,18,20,23,25,30}, e_{1,5,10,12,23,27,28}, e_{8,10,25})$ . Since a fraction of  $2^{-96}$  of these pairs remain, we have about  $2^{13} (= 2^{109} \cdot 2^{-96})$  analyzed pairs. And then, we guess a 64-bit subkey of the  $29^{th}$  and  $30^{th}$  steps and decrypt the analyzed pairs using a guessed key. If a difference of decrypted texts is  $\beta'$ , the counter of a guessed key is increased. Since the signal-to-noise is extremely high, we can distinguish the right subkey in the key space. Thus, the attack requires  $2^{110}$  chosen plaintexts and processing equivalent to about  $2^{14} \cdot 2^{64} \cdot \frac{2}{30} \simeq 2^{75.1}$  30-step SHACAL encryptions.

Also, we can attack on reduced-round SHACAL with at least 160-bit keys using the 30-step differential characteristic  $\alpha \rightarrow \beta''$ . To attack successfully, we must ask for  $2^{140}$  pairs with the input difference  $\alpha$ . The attack procedure is similar to that of reduced-round SHACAL with 128-bit key. Assume that for  $i = 0, 1, 2, 3, 4$ , the reduced  $(32 + i + \theta(i))$ -step SHACAL uses the  $(160 + 32 \cdot i)$ -bit master key. Here the controller  $\theta(i)$  is defined as  $\theta(0) = \theta(1) = 0$ ,  $\theta(2) = \theta(3) = -1$  and  $\theta(4) = -2$ . Since  $E_f$  consists of  $(i + \theta(i) + 2)$  steps, we can find the  $(32 \cdot (i + \theta(i) + 2))$ -bit subkey of the reduced  $(32 + i + \theta(i))$ -step SHACAL. The attack for  $(32 + i + \theta(i))$ -step SHACAL requires  $2^{141}$  chosen plaintexts and processing equivalent to about  $2^{141} \cdot 2^{-32 \cdot (3 - i - \theta(i))} \cdot 2^{32 \cdot (i + \theta(i) + 2)} \cdot \frac{i + \theta(i) + 2}{32 + i + \theta(i)} (\leq 2^{106 + 64 \cdot i + 64 \theta(i)})$   $(32 + i + \theta(i))$ -step SHACAL encryptions where  $i = 0, 1, 2, 3, 4$ . ( $2^{-32 \cdot (3 - i - \theta(i))}$  is a fraction of the analyzed pairs among all of the pairs.) The reason to exist the controller  $\theta(i)$  is that we decrypt only analyzed pairs for a guessed key.

Also, for reduced-round SHACAL with at least 320-bit key, we can attack without the process of filtering out. Assume that for  $j = 0, 1, \dots, 6$ , the reduced  $(35 + j)$ -step SHACAL uses the  $(320 + 32 \cdot j)$ -bit master key. Since  $E_f$  consists of  $(j + 5)$  steps, we can find the  $(32 \cdot (j + 5))$ -bit subkey for the reduced  $(35 + j)$ -step SHACAL. The attack for  $(35 + j)$ -step SHACAL requires  $2^{141}$  chosen plaintexts and processing equivalent to about  $2^{141} \cdot 2^{32 \cdot (j + 5)} \cdot \frac{j + 5}{35 + j} (\leq 2^{299 + 32 \cdot j})$   $(35 + j)$ -step SHACAL encryptions where  $j = 0, 1, \dots, 6$ . Thus, we can attack 41-step SHACAL with 512-bit key.

## 5 Conclusion

SHACAL has short differential characteristics with high probabilities and long ones with low probabilities. From this fact, we could find a 36-step boomerang-distinguisher and attack reduced-round SHACAL with various key sizes. And we discussed the security of reduced-round SHACAL against differential cryptanal-



ysis(DC). In the comparison of an amplified boomerang attack and a differential attack, the latter is more efficient for SHACAL with a 128-bit key, but for SHACAL with other key sizes, the former is more efficient.

## Acknowledgment

We would like to thank referees for their helpful comments.

## References

1. E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
2. E. Biham, O. Dunkelman and N. Keller, *The Rectangle Attack-Rectangling the Serpent*, Proc. of Eurocrypt'2001, Springer-Verlag, LNCS 2045, pp.340-357, 2001
3. H. Handschuh, D. Naccache, *SHACAL*, In Proceedings of the First Open NESSIE Workshop, November 2000.
4. H. Handschuh, D. Naccache, *SHACAL*, NESSIE project, October 2001.
5. H. Handschuh, L. R. Knudsen, and M. J. Robshaw *Analysis of SHA-1 in Encryption Mode*, CT-RSA 2001, Springer-Verlag, LNCS 2020, pp.70-83, 2001.
6. J. Kelsey, T. Kohno, and B. Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, Proc. of FSE'2000, Springer-Verlag, LNCS 1978, pp.75-93, 2001
7. J. Nakahara Jr, *The Statistical Evaluation of the NESSIE Submission*, October 2001.
8. David Wagner, *The boomerang Attack*, proceedings of Fast Software Encryption, Lecture Notes in Computer Science 1636, pp.156-170, Springer-Verlag, 1999.

# Enhancing Differential-Linear Cryptanalysis<sup>\*</sup>

Eli Biham<sup>1</sup>, Orr Dunkelman<sup>1</sup>, and Nathan Keller<sup>2</sup>

<sup>1</sup> Computer Science Department, Technion, Haifa 32000, Israel,  
{biham, orrd}@cs.technion.ac.il

<sup>2</sup> Mathematics Department, Technion, Haifa 32000, Israel,  
nkeller@tx.technion.ac.il

**Abstract.** Differential cryptanalysis analyzes ciphers by studying the development of differences during encryption. Linear cryptanalysis is similar but is based on studying approximate linear relations. In 1994, Langford and Hellman showed that both kinds of analysis can be combined together by a technique called *differential-linear cryptanalysis*, in which the differential part creates a linear approximation with probability 1. They applied their technique to 8-round DES. In this paper we present an enhancement of differential-linear cryptanalysis in which the inherited linear probability is smaller than 1. We use this extension to describe a differential-linear distinguisher for a 7-round reduced-version of DES, and to present the best known key-recovery attack on a 9-round reduced-version of DES. We use our enhanced technique to attack COCONUT98 with time complexity  $2^{33.7}$  encryptions and  $2^{27.7}$  chosen plaintexts.

## 1 Introduction

Differential cryptanalysis [2] analyzes ciphers by studying the development of differences during encryption. Linear cryptanalysis [11] is similar but is based on studying approximate linear relations.

In 1994, Langford and Hellman [10] showed that both kinds of analysis can be combined together by a technique called *differential-linear cryptanalysis*, in which the differential part creates a linear approximation with probability 1. Using their new technique they have succeeded to analyze up to 8-round reduced variants of DES [12] using only 512 chosen plaintext in a few seconds on a personal computer. This attack is so far the best known attack on 8-round DES [1].

The differential-linear technique was later applied to analyze the IDEA cipher [9]: a reduced version of IDEA was analyzed by a differential-linear attack in [6], and differential-linear weak keys of the full IDEA (along with a related-key differential-linear attack on reduced IDEA) were found in [7]. It was also shown that the ciphertext-only extension of differential and linear cryptanalysis works also with differential-linear cryptanalysis [5].

---

<sup>\*</sup> The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-1999-12324.

<sup>1</sup> From now on we will use the shorthand *r-round DES* for an *r*-round reduced version of DES.

Langford and Hellman’s technique is an example for devising the “distinguisher” used in the attack as a combination of two much simpler parts; in this case a combination of a differential characteristic and a linear approximation. Such combinations were later used in other kinds of cryptanalysis, e.g., cryptanalysis using impossible differentials [4,3] (miss in the middle), and boomerang attacks [15], both use combinations of differential characteristics.

In this paper we present an extension of differential-linear cryptanalysis in which the linear probability induced by the differential characteristic is smaller than 1. We use this extension to describe a differential-linear distinguisher for 7-round DES, and then present a differential-linear key-recovery attack on 8-round and 9-round DES. This extension can attack DES with up to 10 rounds, where the 9-round variant of the attack is by far the best known attack against 9-round DES. We also apply the technique to the full COCONUT98.

This paper is organized as follows: In Section 2 we describe Langford and Hellman’s differential-linear attacks. In Section 3 we present our differential-linear extension. In Section 4 we present the distinguishing attack on 7-Round DES. In Sections 5 and 6 we present the key recovery attacks on 8-Round and 9-Round DES, respectively. In Section 7 we present a key recovery attack on COCONUT98. Finally, Section 8 summarizes the paper.

## 2 Differential-Linear Cryptanalysis

Langford and Hellman [10] show that a concatenation of a differential characteristic and a linear characteristic can be performed. They select a 3-round characteristic of DES, which predicts the differences of a few bits after three rounds with probability 1 (the probability for the whole block difference after three rounds is much lower). So, given a pair of plaintexts with the required plaintext difference, they know the difference of a few bits after three rounds for certain. They use a 3-round linear approximation for rounds 4–6. If the difference in the intermediate data before the linear approximation can be predicted, then we can obtain information about the parities. More precisely, if the difference in the input subset can be predicted, then we know whether the input subset parity in both encryptions is the same or differ. As the linear approximation predicts the output subset parity, we can now predict whether the output subset parities of the two ciphertexts are more likely to be the same or not. Fortunately, they found differential and linear characteristics in which the subset required for the parity is predicted with probability 1 by the differential characteristic. Thus, the differential characteristic actually tells them the difference of the two parities. Both difference and parity are linear operations (they both use XOR). Thus, the two linear approximations in rounds 4–6 in both encryptions can be combined into a six-round approximation of rounds

$$6_1-5_1-4_1\text{-differential-}4_2-5_2-6_2,$$

where the subscript denote whether the round is in the first encryption or the second, and “differential” refers to the differential combiner that ensures that

the parities of the data before round 4 in both encryptions are always equal (or always differ).

This enlarged linear characteristic has twice as many rounds as the original, thus its probability is much closer to  $1/2$  than the original. However, it is still usable, and in various cases it leads to the best known attacks against the analyzed cipher, as in the case of the differential-linear attack on 8-round DES described by Langford and Hellman.

The differential-linear distinguisher is based on encrypting many pairs with some known input difference. Each pair is encrypted, and the output subset parity is computed for both ciphertext. The fraction of times when the two parities agree differ from  $1/2$  for a good differential-linear characteristic. Thus, it can be used to distinguish the cipher from a random permutation. A key recovery attack can be mounted using standard techniques (guessing the following round subkey, etc.).

### 3 Our Differential-Linear Extension

We observed that in the above approximation

$$6_1-5_1-4_1\text{-differential-}4_2-5_2-6_2$$

all the rounds are approximated with probabilities which may be different than  $1/2 \pm 1/2$ , except for the connection by the differential characteristic, which has probability 1.

From now on, we use notations based on [12] for differential and linear cryptanalysis, respectively. In our notations  $\Omega_P$ ,  $\Omega_T$  are the input and output differences of the differential characteristic, and  $\lambda_P$ ,  $\lambda_T$  are the input and output subsets (denoted by bit masks) of the linear characteristic.

In this paper we propose using a differential connection with fractional probabilities. Let the probability of the linear characteristic be denoted by  $1/2 + q$ , and the probability of the differential characteristic be denoted by  $p'$  (in the case of Langford and Hellman,  $p' = 1$  and  $q = 0.195$ ).

Given the probabilities of the differential characteristic  $p'$ , we approximate the linear probability  $1/2 + p$  of the relations of the parities of the subsets of bits  $\lambda_P$  between both encryptions (in the particular case of  $p' = 1$  certainly  $p = 1/2$  as in Langford and Hellman's analysis), and then compute the probability of the total relation (from the last round of the linear characteristic through all its rounds backward in the first encryption, through the differential approximation of the parity, through the linear characteristic in the second encryption to its last round). This probability is computed by the usual rule for probabilities of concatenation of linear characteristics. Thus, the total probability is

$$1/2 + 4pq^2.$$

Note that the differential probability  $p'$  is the probability for the expected difference in the required subset of bits, which is usually different (higher) than

the probability of the differential characteristic with the full block output difference, so the best characteristic for our purposes may be different than the best ordinary characteristic. For example, in Langford and Hellman's case the characteristic predicts with probability 1 only 36 bits of the 64 bits of the output difference; these 36 bits include all the 5 bits of  $\lambda_P$ . Given an ordinary differential characteristic with probability  $p_\Omega$ , we know that the full block output difference  $\Omega_T$  appears with probability  $p_\Omega$ , and that with probability  $1 - p_\Omega$  the difference is different. When considering only the subset of bits in  $\lambda_P$ , the probability of the characteristic on these bits becomes  $p'$ . The probability  $1/2 + p$  can now be approximated by

$$1/2 + p \approx p' + (1 - p')/2 = 1/2 + p'/2,$$

assuming that the parity in the rest of the cases is uniformly distributed. This assumption is not necessarily accurate, for example, there might be other high-probability differential characteristics with the same plaintext difference, but with different (or same) parity of the subset of bits of the difference. Thus, this approximated probability should be verified by the designer of an attack, and if possible, he should perform a more accurate computation of the probability, or check it experimentally.

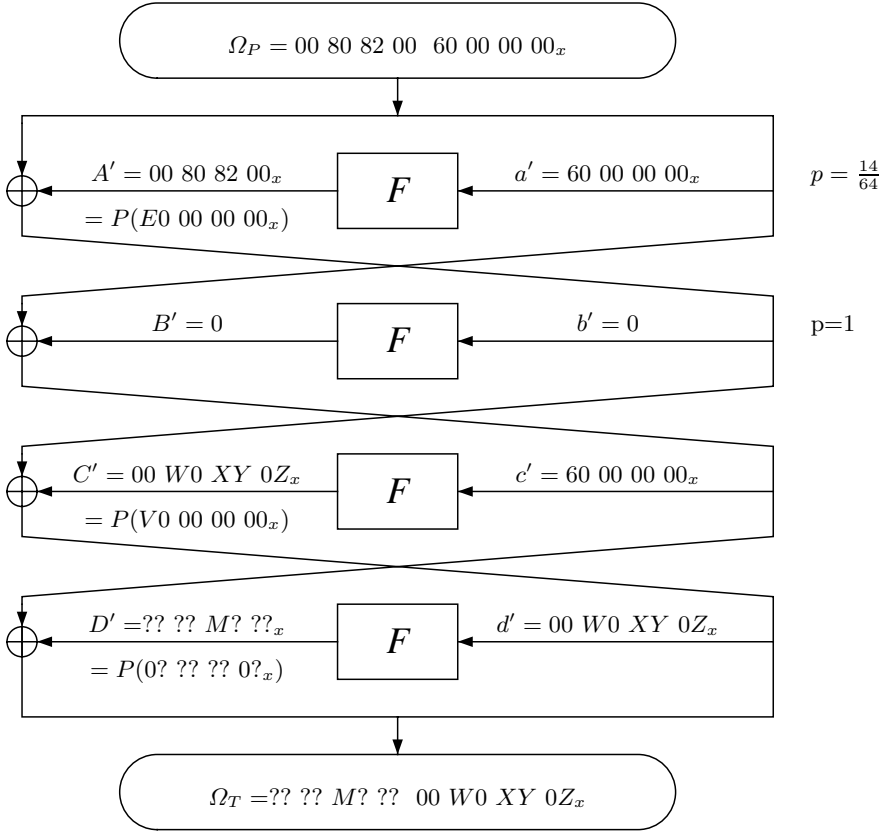
It is worth mentioning that the attack works even if the differential characteristic predicts that there are differences in some of the bits in the subset  $\lambda_P$ . All we need is to know the parity of the differences of the bits in  $\lambda_P$  (rather than fixing the differences to 0). For example, assume that  $\Omega_T = 10\ ?0\ 67\ 80\ D7\ 6?\ 11_x$  (where the ? denotes an unpredicted hex digit) and assume that  $\lambda_P = 00\ 00\ 08\ D7\ 00\ 00\ 00\ 01_x$ . Then, the 8 bits selected by  $\lambda_P$  are known in  $\Omega_T$ , of which 5 have value 1 and 3 have value 0. Therefore, the expected parity of the differences of the two runs is  $\Omega_T \cdot \lambda_P = 1$ . Note that even if  $\Omega_T \cdot \lambda_P$  is unknown but constant, the attack still succeeds.

## 4 A Distinguishing Attack on 7-Round DES

We now present an attack that distinguishes whether a cipher (given in a form of a black box) is a 7-round DES, or a random permutation, using the differential-linear technique. We use the following 4-round extended differential characteristic with probability  $p_\Omega = 14/64 = 0.21875$ , which is an extension by one round of the 3-round characteristic used by Langford and Hellman. This extended characteristic is presented in Figure 11.

The 3-round differential was concatenated with the 3-round linear approximation with probability  $1/2 + 0.195$  presented in Figure 12. This 3-round linear approximation is also the best 3-round linear approximation for DES.

We use our 4-round differential characteristic to build a distinguisher with a probability of  $1/2 + p \approx 1/2 + \frac{14/64}{2} \approx 1/2 + 0.109$  (recall, that for a random permutation this value is  $1/2$ ). This approximation assumes that the behavior of the remaining fraction of  $1 - 14/64 = 50/64$  of the pairs induces uniform linear distribution. We have verified the value of  $p$  experimentally, and confirmed this



(where  $V \in \{1, \dots, F_x\}$ ,  $W \in \{0, 8\}$ ,  $X \in \{0, 8\}$ ,  $Y \in \{0, 2\}$ ,  $Z \in \{0, 2\}$ ,  $M \in \{0, \dots, 7\}$ , and any  $?$  is any arbitrary value)

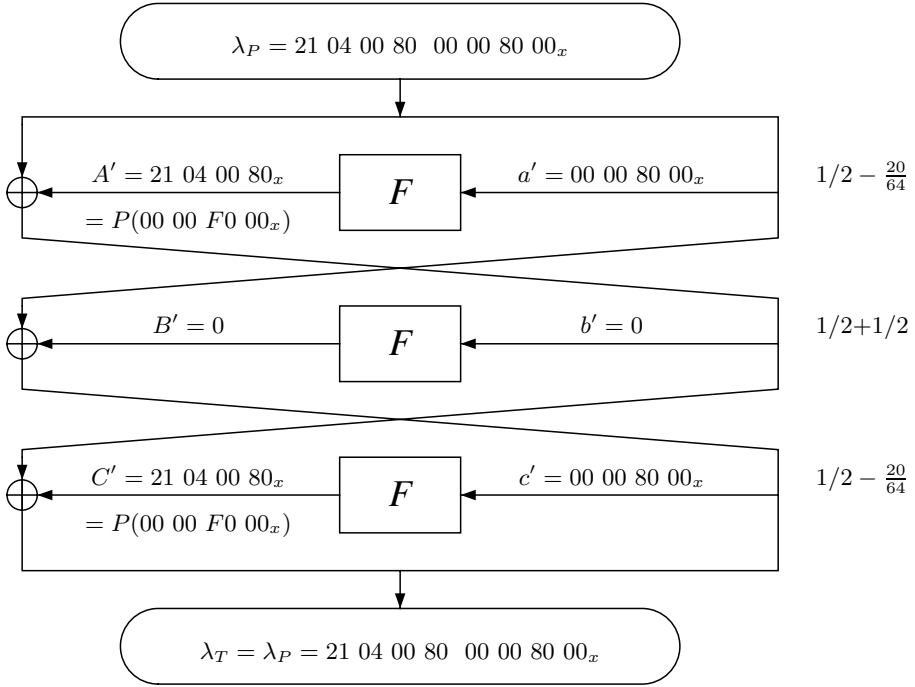
**Fig. 1.** The Extended 4-Round Differential Characteristic Used in Our Distinguisher

probability using hundreds of different keys, and millions of encrypted pairs. The linear characteristic has probability  $1/2 + q = 1/2 + 2(\frac{-20}{64})^2 \approx 1/2 + 0.195$ . The total probability of the approximation is thus

$$1/2 + 4pq^2 = 1/2 + 4 \cdot 0.109 \cdot 0.195^2 = 1/2 + 0.0167 = 1/2 + 2^{-5.91}.$$

The distinguishing attack is as follows:

1. Select  $N = 2^{11.81}$  plaintext pairs with the plaintext difference  $\Omega_P = 00\ 80\ 82\ 00\ 60\ 00\ 00\ 00_x$ .
2. Request the ciphertexts of these plaintext pairs (encrypted under the unknown key  $K$ ).
3. For each ciphertext pair, compute the parity of the bits masked by  $\lambda_T = 21\ 04\ 00\ 80\ 00\ 00\ 80\ 00_x$  in each of the plaintexts, and count for how many pairs both parities are equal. Let the number of such pairs be denoted by  $m$ .



**Fig. 2.** The 3-Round Linear Approximation Used in [10]

4. If

$$\frac{m}{N} > \frac{1}{2} + \epsilon, \quad \text{where } \epsilon = \frac{4pq^2}{2} = 2^{-6.90},$$

(i.e.,  $m > 2^{10.81} + 2^{4.91}$ ) conclude that the cipher is 7-round DES.

5. Otherwise, conclude that the cipher is not 7-round DES.

The parameters  $N$  and  $\epsilon$  are selected as to maximize the success rate of the attack while requiring the lowest data complexity. For  $N = 2^{11.81}$  and  $\epsilon = 2^{-6.90}$  the attack succeeds with probability higher than 84.13%, and has data and time complexities of  $2^{12.81}$ .

We point out that unlike most linear attacks (and most differential-linear attacks) it suffices in this case to test whether  $\frac{m}{N} > \frac{1}{2} + \epsilon$  rather than whether  $|\frac{m}{n} - 1/2| \geq \epsilon$ . This follows from the fact that in this specific attack the bias is always positive and is unaffected by any key bit (as all the affected key bits are used twice and thus cancelled).

In order to show that for these parameters we get this success rate we use the following statistical reasoning (see [13]): For a random permutation each pair behaves randomly, and thus in half of the pairs the two parities of the subset of the ciphertext bits are equal. Therefore, the number of equal parities behaves like a binomial random variable  $X \sim \text{Bin}(2^{11.81}, 1/2)$ . It is easy to see that such random variable can be approximated according to the normal distribution, and

thus we conclude that the probability that this random variable (counting the number of pairs with equal parities) is higher than  $2^{10.81} + 2^{4.91}$  is at most 15.87%. We conclude that for a random permutation the probability that the above algorithm outputs ‘this is a random permutation’ is 84.13%.

Repeating this analysis for 7-round DES with  $X \sim \text{Bin}(2^{11.81}, 1/2 + 2^{-5.91})$  the probability that the algorithm outputs ‘this is a random permutation’ is 15.87%.

## 5 A Key Recovery Attack on 8-Round DES

This attack can be extended to a key-recovery attack by adding one round for the analysis, and using the 7-round distinguisher, as follows

1. Select  $N = 2^{13.81}$  plaintext pairs with the plaintext difference  $\Omega_P = 00\ 80\ 82\ 00\ 60\ 00\ 00\ 00_x$ .
2. Request the ciphertexts of these plaintext pairs (encrypted under the unknown key  $K$ ).
3. Initialize an array of 64 counters to zeroes.
4. For each ciphertext pair
  - (a) Try all the 64 possible values of the 6 bits of the subkey  $K_8$  that enter the S Box  $S_1$  in round 8.
  - (b) For each value of the subkey, compute the output of  $S_1$  in the last round, and use its output to compute the parity of the subset of bits in  $\lambda_T$  after round 7. Now we can compute the output subset parity, as we know 4 of the subset bits from the ciphertext, and the remaining one from the the output of  $S_1$  and the ciphertext.
  - (c) If the parities in both members of the pair are equal, increment the counter in the array which relates to the 6 bits of the subkey.
5. The highest entry in the array should correspond to the six bits of  $K_8$  entering  $S_1$  in round 8.
6. The rest of the key bits can be recovered by auxiliary techniques.

For  $N = 2^{13.81}$  this attack succeeds with probability 77.27% or more. The complexity of the attack is  $2^{14.81} \cdot 2^6/64 = 2^{14.81}$  time (in units of 8-round DES encryptions;  $2^6$  subkeys tried, each trial takes about one S box computation out of the 64 S boxes of a full encryption), requiring  $2^{14.81}$  chosen plaintexts.

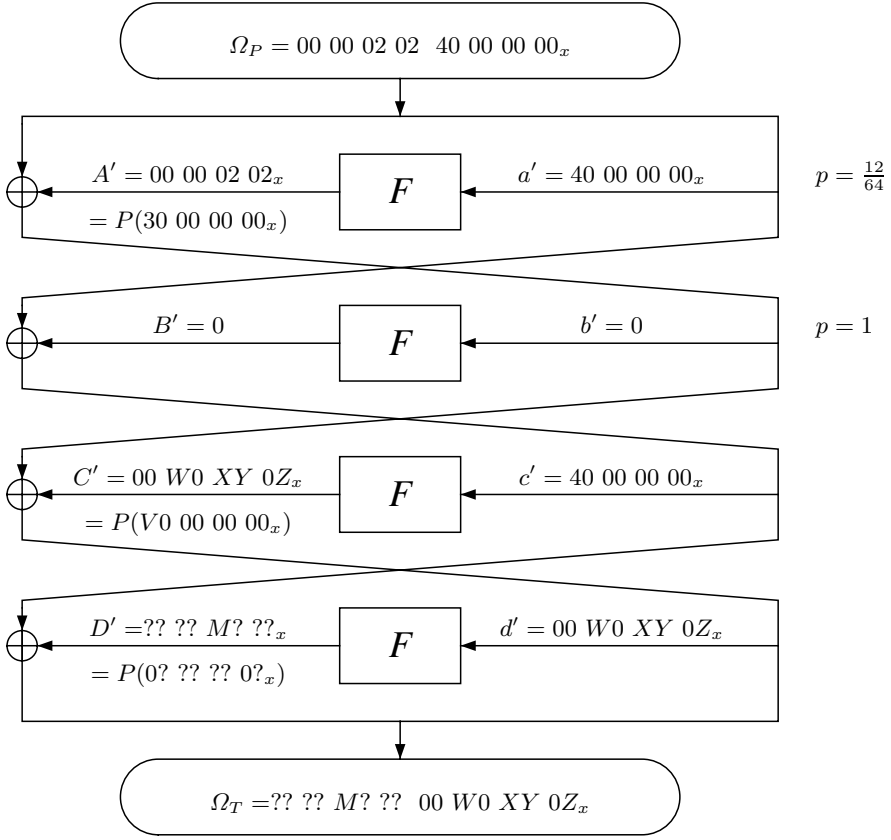
## 6 A Key Recovery Attack on 9-Round DES

Similarly, the attack can be extended to 9 rounds by analyzing two rounds in addition to the 7-round distinguisher.

In this case we use the slightly modified differential characteristic presented in Figure 3

This characteristic is similar to the original, except that its first round is replaced. This replacement is done to reduce the number of active S boxes in the





**Fig. 3.** The Modified 4-round Differential Characteristic Used in the 9-Round Attack

round preceding the characteristic. This characteristic induces a linear probability of  $1/2 + p = 1/2 + 0.09375$  (again, we experimentally verified this probability). With this change, the 7-round distinguisher with 84.13% success rate would require  $N = 2^{12.25}$  pairs (for  $\epsilon = 2^{-7.13}$ ).

To mount a 9-round key recovery attack, we set the differential and linear characteristics combination at rounds 2–8, and analyze rounds 1 and 9.

The attack is as follows

1. Select  $N = 2^{15.75}$  plaintexts, consisting of  $2^{6.75}$  structures, each is chosen by selecting:
  - (a) Any plaintext  $P_0$
  - (b) The plaintexts  $P_1, \dots, P_{255}$  which differ from  $P_0$  by all the 255 possible subsets of the eight bits masked by  $18\ 22\ 28\ 28\ 00\ 00\ 00\ 00_x$  (these are the output bits of S6 and S8 in round 1).
  - (c) The plaintexts  $P_{256}, \dots, P_{511}$  selected as  $P_i = P_{i-256} \oplus 40\ 00\ 00\ 00\ 00\ 00\ 02\ 02_x$ .

2. Request the ciphertexts of these plaintext pairs (encrypted under the unknown key  $K$ ).
3. At this stage we do not know which pairs in the structure have the difference  $\Omega_P$  before round 2. Instead, we guess these pairs by trying all the possible values of the 12 bit of the subkey  $K_1$  which enter  $S_6$  and  $S_8$ .
4. For each value of the 12 bits of  $K_1$  entering  $S_6$  and  $S_8$ 
  - (a) Partially encrypt  $S_6$  and  $S_8$  in the first round of each plaintext and find the pairs which satisfy the difference  $\Omega_P$  before round 2 (assuming the guessed value is correct)
  - (b) Given all the pairs, apply the 8-round attack on these pairs (the attack is on the 8 rounds from round 2 to round 9).
5. Each trial of the key gives us  $12 + 6 = 18$  bits of the subkeys (12 bits in round 1 and 6 bits in round 9), along with a measure for correctness (which is the number of times it is suggested in the 8-round attack). The correct value of the 18 bits is expected to be the most frequently suggested values (with over 88.80% success rate).
6. The rest of the key bits are then recovered by auxiliary techniques.

Note that due to the mass of  $2^{12}$  applications of the 8-round attack, and the need to identify which application uses the correct guess of the 12 bits of the first subkey, we need more data than for a single application of the 8-round attack.

This attack requires  $2^{15.75}$  chosen plaintexts, and finds the key in time  $2^{15.75} \cdot 2^{12} \cdot 2^6 \cdot 3/72 \approx 2^{29.17}$  (in units of 9-round DES encryptions). This time complexity of this attack can be further reduced using auxiliary techniques and reordering of the operations.

## 7 Attack on COCONUT98

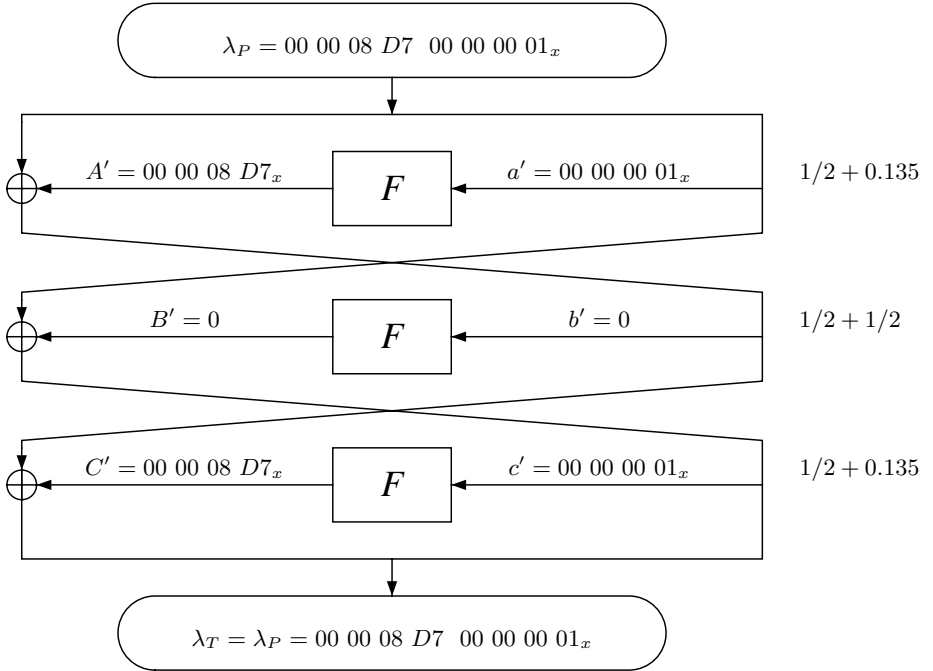
We can use our results to present the best known attack against the COCONUT98 block cipher. COCONUT98 is a 64-bit blocksize 256-bit keysize block cipher, that was designed using the decorrelation theory [14].

The cipher is composed of 4 Feistel rounds, a decorrelation module, and 4 additional Feistel rounds. The decorrelation module is  $M(xy) = (xy \oplus K_5 K_6) \times K_7 K_8 \bmod GF(2^{64})$ , where  $x, y$  are the 32-bit data word,  $xy$  denotes their concatenation.  $K_5, K_6, K_7, K_8$  are four 32-bit values supplied by the user key, and where  $K_7 K_8 \neq 0$ . The multiplication is over the finite field  $GF(2^{64})$  defined by the polynomial  $x^{64} + x^{11} + x^2 + x + 1$  over  $GF(2)$ . Note that the exact underlying polynomial has no effect on our results. The Feistel rounds can be described as follows:

$$\begin{aligned}\phi(x) &= x + 256 \cdot S(x \bmod 256) \bmod 2^{32} \\ F_{k_i}(x, y) &= (y, x \oplus \phi(ROL_{11}(\phi(y \oplus k_i)) + c \bmod 2^{32}))\end{aligned}$$

where  $c = B7E15162_x$  is a known constant and  $k_i$  is the round subkey.

In [15] a 4-round differential (of the Feistel rounds) with probability  $0.83 \cdot 2^{-4}$  was introduced. It was commented that the expected difference that enters the



**Fig. 4.** A 3-round Linear Approximation of COCONUT98 with Probability  $1/2+0.0364$

decorrelation module leads to some fixed but unknown difference after the decorrelation module. We denote this differential by  $\Omega_{COCONUT98}$ . Thus,  $\Omega_{COCONUT98}$  is a differential with probability  $p = 0.83 \cdot 2^{-4}$  for the first 4 Feistel rounds and the decorrelation module. Note that we have no idea what the output difference of  $\Omega_{COCONUT98}$  is. Still, this does not interfere with our analysis, as we mentioned before. In case that the subset  $\lambda_P$  of bits of this output difference has an odd number of active bits (i.e., the scalar product  $\lambda_P \cdot \Omega_T$  is 1), then there are going to be more disagreements on the output parity than agreements, and the linear bias would be negated, without affecting the analysis.

In Figure 4 we present a linear approximation for 3 Feistel rounds of COCONUT98. This approximation has a probability  $1/2 + q = 1/2 + 0.0364$ .

We can now use  $\Omega_{COCONUT98}$  concatenated to the 3-round linear approximation to present a distinguisher for the entire COCONUT98 but the last round. The distinguisher has a bias of  $4pq^2 \approx 1/3638$ . Note that we do not know whether the bias is in favor of having the same parity, or having complement parities (as we have no idea what the output of the differential is; this output depends on some key that we do not know), but this does not stop us from attacking the cipher.

The attack retrieves subkey bits of the last round. As the only unknown value in the equation of the parities is the least significant bit of the right half after the 7th Feistel round (just after the approximation), we need to determine the

least significant bit of the output of the function  $F$  in the last round. As this bit is unaffected by the second  $\phi$ , and as the addition of the constant  $c$  does not change it (the least significant bit of the constant  $c$  is 0), then it is unchanged after the rotate left operation. In this operation we actually need to know bit 21 before the rotate. In order to determine this bit, we need to know the lower 22 bits that enter the first  $\phi$ , and we conclude that we need to know the 22 lower key bits in the last round. As guessing these 22 key bits can be very time consuming, we try to look for more efficient solutions. We can approximate (with very high probability) the true value of the relevant bit, by knowing the output of the S-box in the first  $\phi$  (i.e., look at the 8 lower subkey bits) and  $m$  bits of the subkey from bit 21 and downward. Considering only  $m$  subkey bits causes a mistake in a fraction of  $2^{-m}$  of the cases. As this mistake appears uniformly and affects all trials similarly, we actually get a bias of  $4pq^2 \cdot (1 - 2^{-m+1})$ . For the value  $m = 7$  this bias is  $1/3700$ . A slight improvement of the bias to the value  $4pq^2 = 1/3638$  can be performed by discarding some mistaken data.

Our attack counts over these  $7+8 = 15$  subkey bits using the following algorithm:

1. Initialize  $2^{15}$  counters. Each corresponds to a different last round subkey.
2. Encrypt  $N$  pairs with the required input difference.
3. For each 15-bit subkey value partially decrypt all ciphertext pairs and check whether the parities of the subsets are equal or different. For each ciphertext pair increment the counter of the subkey in case of equality.
4. Look for the counter with the maximal bias from  $N/2$  (i.e.,  $|\text{counter} - N/2|$  is maximal), and suggest the related subkey as the right subkey.

The time complexity of this algorithm is  $2N$  encryptions and  $2 \cdot 2^{15} \cdot N$  additional last round activations (and  $2^{16}$  additional memory accesses, which we omit). Hence, the total running time of the algorithm is  $2^{16} \cdot N/8 = 2^{13} \cdot N$  COCONUT98 encryptions.

We now determine  $N$ . We associate the right key counter with the random variable  $X$ , and each of the  $2^{15} - 1$  wrong subkeys with its own random variable  $Y_i$ . We assume that all of these variables have a normal distribution and that  $X \sim N(N/2 + N/3700, N/4)$  and that  $\forall i : Y_i \sim N(N/2, N/4)$ . For  $N = 8/(1/3700)^2$ , the success rate of the attack is at least 75.46%. Thus, we conclude that we need  $N = 8 \cdot 3700^2 = 2^{26.7}$  pairs ( $2^{27.7}$  chosen plaintexts), and time complexity of  $2^{39.7}$  COCONUT98 encryptions.

The rest of the key can be found with auxiliary techniques using other differentials and linear approximations with a negligible additional time and data complexities.

We can reduce the time complexity of the attack by observing that we are actually interested in 15 bits of the ciphertext. In the above analysis we perform the same operations for the same values many times. Using a precomputed table (which requires  $2^{15} \cdot 2^{15} = 2^{30}$  last round activations to compute) we can reduce the time complexity of the attack to  $2^{39.7}$  memory accesses, which are equivalent to at most  $2^{33.7}$  COCONUT98 encryptions.

## 8 Summary and Conclusions

In this paper we presented an extension of differential-linear cryptanalysis that allows using a differential characteristic with probability lower than 1. We showed that this extension can attack DES reduced 7, 8, and 9 rounds. The latter is the best known method against 9-round DES.

This attack can be extended to analyze the 10-round reduced-variant of DES with time complexity about  $2^{50}$  and using about  $2^{20}$  chosen plaintexts.

We also presented the fastest attack on the full COCONUT98. Our attack requires about  $2^{27.7}$  chosen plaintexts and time complexity of about  $2^{33.7}$  COCONUT98 encryptions. Previous results [15] required  $2^{16}$  adaptive chosen plaintexts and ciphertexts and  $2^{38}$  COCONUT98 encryptions.

We summarize our results along with previously known results in Table 1.

**Table 1.** Summary of Our Results and Previously Known Results

Cipher	Attack	Complexity		Success
		Data	Time	Rate
8-round DES	Differential [2]	$2^{14}$ CP	$2^9$	53%
	Linear [11]	$2^{18}$ KP	$2^{25}$	49.4%
	Linear [11]	$2^{19}$ KP	$2^{26}$	93.2%
	Differential-Linear [10]	512CP	$2^{14}$	80%
	Differential-Linear [10]	768CP	$2^{14.6}$	95%
	C.P. Linear Cryptanalysis [8]	$2^{16}$ CP	$2^{23}$	51%
	C.P. Linear Cryptanalysis [8]	$2^{17}$ CP	$2^{24}$	94%
	Enhanced Differential-Linear – this paper	$2^{14.8}$ CP	$2^{14.8}$	77.3 %
9-round DES	Differential [2]	$2^{24}$ CP	$2^{32}$	99.97%
	Enhanced Differential-Linear – this paper	$2^{15.8}$ CP	$2^{29.2}$	88.8%
COCONUT'98	Boomerang [15]	$2^{16}$ ACPC	$2^{38}$	99.96%
(full cipher)	Enhanced Differential-Linear – this paper	$2^{27.7}$ CP	$2^{33.7}$	75.5%

Complexity is measured in encryption units.

CP - Chosen Plaintexts, KP - Known Plaintexts

ACPC - Adaptive Chosen Plaintexts and Ciphertexts

## References

1. Biham Eli, *On Matsui's Linear Cryptanalysis*, Advances in Cryptology, proceedings of EUROCRYPT '94, Lecture Notes in Computer Science 950, pp. 341–355, 1994.
2. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
3. Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 124–138, 1999.

4. Eli Biham, Alex Biryukov, Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*, Advances in Cryptology, proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pp. 12–23, 1999.
5. Alex Biryukov, Eyal Kushilevitz, *From Differential Cryptoanalysis to Ciphertext-Only Attacks*, Advances in Cryptology, proceedings of CRYPTO '98, Lecture Notes in Computer Science 1462, pp. 72–88, 1998.
6. Johan Borst, Lars R. Knudsen, Vincent Rijmen, *Two Attacks on Reduced Round IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '97, Lecture Notes in Computer Science 1233, pp. 1–13, 1997.
7. Philip Hawkes, *Differential-Linear Weak Keys Classes of IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '98, Lecture Notes in Computer Science 1403, pp. 112–126, 1998.
8. Lars R. Knudsen, John Erik Mathiassen, *A Chosen-Plaintext Linear Attack on DES*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science 1978, pp. 262–272, 2001.
9. Xuejia Lai, James L. Massey, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology, proceedings of EUROCRYPT '91, Lecture Notes in Computer Science 547, pp. 17–38, 1992.
10. Suzan K. Langford, Martin E. Hellman, *Differential-Linear Cryptanalysis*, Advances in Cryptology, proceedings of CRYPTO '94, Lecture Notes in Computer Science 839, pp. 17–25, 1994.
11. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology, proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pp. 386–397, 1994.
12. US National Bureau of Standards, *Data Encryption Standard*, Federal Information Processing Standards Publications No. 46, 1977.
13. Jim Pitman, *Probability*, Springer-Verlag, 1993.
14. Serge Vaudenay, *Provable Security for Block Ciphers by Decorrelation*, proceedings of STACS '98, Lecture Notes in Computer Science 1373, pp. 249–275, 1998.
15. David Wagner, *The Boomerang Attack*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 156–170, 1999.

# Cryptanalysis of Block Ciphers with Overdefined Systems of Equations

Nicolas T. Courtois<sup>1,\*</sup> and Josef Pieprzyk<sup>2</sup>

<sup>1</sup> CP8 Crypto Lab, SchlumbergerSema,  
36-38, rue de la Princesse, BP 45, 78430 Louveciennes Cedex, France,  
[courtois@minrank.org](mailto:courtois@minrank.org)

<sup>2</sup> Center for Advanced Computing – Algorithms and Cryptography,  
Department of Computing, Macquarie University, Sydney, NSW 2109, Australia,  
[josef@ics.mq.edu.au](mailto:josef@ics.mq.edu.au)

**Abstract.** Several recently proposed ciphers, for example Rijndael and Serpent, are built with layers of small S-boxes interconnected by linear key-dependent layers. Their security relies on the fact, that the classical methods of cryptanalysis (e.g. linear or differential attacks) are based on probabilistic characteristics, which makes their security grow exponentially with the number of rounds  $N_r$ .

In this paper we study the security of such ciphers under an additional hypothesis: the S-box can be described by an overdefined system of algebraic equations (true with probability 1). We show that this is true for both Serpent (due to a small size of S-boxes) and Rijndael (due to unexpected algebraic properties). We study general methods known for solving overdefined systems of equations, such as XL from Eurocrypt'00, and show their inefficiency. Then we introduce a new method called XSL that uses the sparsity of the equations and their specific structure.

The XSL attack uses only relations true with probability 1, and thus the security does not have to grow exponentially in the number of rounds. XSL has a parameter  $P$ , and from our estimations it seems that  $P$  should be a constant or grow very slowly with the number of rounds. The XSL attack would then be polynomial (or subexponential) in  $N_r$ , with a huge constant that is double-exponential in the size of the S-box. The exact complexity of such attacks is not known due to the redundant equations. Though the presented version of the XSL attack always gives always more than the exhaustive search for Rijndael, it seems to (marginally) break 256-bit Serpent. We suggest a new criterion for design of S-boxes in block ciphers: they should not be describable by a system of polynomial equations that is too small or too overdefined.

**Key Words:** Block ciphers, AES, Rijndael, Square, Serpent, Camellia, multivariate quadratic equations, MQ problem, overdefined systems of multivariate equations, XL algorithm, Gröbner bases, sparse multivariate polynomials, Multivariate Cryptanalysis.

---

\* Partially written when visiting the Department of Computing, Macquarie University in Sydney, Australia.

# 1 Introduction

On October 2nd, 2000, NIST has selected Rijndael as the Advanced Encryption Standard. Serpent was second in the number of votes [1].

In the famous paper from 1949, Claude E. Shannon states that breaking a good cipher should require “as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type”, see [25]. This seemed very easy to achieve so far, as solving systems of equations can become intractable very easily. Though every cipher can be described in terms of solving multivariate equations over  $GF(2)$ , it does not mean that it can be broken. In [8] the whole AES is represented by one single equation with  $2^{50}$  terms. Such a big equation has undoubtedly no consequences whatsoever on the security of AES. Recently however surprising attacks appeared in public key cryptography: the cryptanalysis of Matsumoto-Imai cryptosystem [17] by Patarin and the attack on the basic version of HFE cryptosystem by Courtois [6]. In these attacks the security collapses suddenly after discovering the existence of additional multivariate equations, that are not obvious and have not been anticipated by the designers. The subject of this paper is to see if such a weakness can compromise the security of a block cipher. For example, we show that the cryptanalysis of Rijndael and Serpent reduces to solving a big system of Multivariate Quadratic equations (a.k.a. MQ problem). Unlike in [8], MQ is a problem already known in cryptography that underlies the security of multivariate public key schemes such as HFE [19]. In [22,23] Shamir *et al.* show that though MQ is NP-hard, its complexity drops substantially when the MQ becomes overdefined (more equations than unknowns)<sup>1</sup>. In this paper we show that if the MQ is sparse and have a regular structure, it becomes even much easier. Such will be the MQ systems we will write for Rijndael and Serpent ciphers.

Since the pioneering work of Luby-Rackoff [12], there were many developments on the security of top-level schemes of block ciphers. The state of art in both security proofs and generic attacks for Feistel ciphers can be found in [15] and [18]. However, Rijndael is not a Feistel cipher and a more powerful theory has been developed by Vaudenay [26], to make security proofs against a large class of attacks including linear and differential cryptanalysis, for an arbitrary type of cipher. From this theory Moriai and Vaudenay developed security proofs for idealized versions of several AES candidates [27]. The outcome for Rijndael was somewhat strange: the cipher should have  $\geq 384$  rounds in order to make sure it was secure. Similar results were obtained for Serpent. Therefore, it is not completely unreasonable to believe, that the structure of Rijndael and Serpent could allow attacks with complexity growing slowly with the number of rounds. In this paper, it seems that we have found such an attack. It depends however more on algebraic properties of the S-boxes than on the structure of the cipher, and potentially, it can probably be extended to any block cipher.

The paper is organized as follows: First we describe a general class of ciphers that includes Rijndael and Serpent. Then we explore algebraic properties of

---

<sup>1</sup> Solving MQ in the opposite case of **under**defined systems, has been studied in [5].



their S-boxes and show that they can be described by an overdefined system of equations. Consequently, we formulate their cryptanalysis in terms of solving an overdefined system of quadratic equations. Though the general XL attack fails, we will present an improved method called XSL. It does not yet break Rijndael or Serpent but it gives definite conclusions about the design of block ciphers.

## 2 Substitution-Affine Ciphers, Rijndael and Serpent

According to Shannon's paradigm [25], a cipher is combination of some confusion and diffusion components. For example, SP-networks [710] are combinations of S-boxes with permutations of bits. More generally, we may allow linear or affine functions of bits, not only permutations of wires. We will call it a SA-cipher. In [21] Shamir and Biryukov study general top-level structural attacks against the SA-ciphers. These attacks will not depend on particular S-boxes used.

In the present paper we use specific properties of the S-boxes. We specify a restricted class of SA-ciphers called XSL-ciphers. Though our attacks are designed for XSL-ciphers, it is obvious that they can be easily extended to all SA-ciphers, and even to other block ciphers (including Feistel ciphers), provided that they use (only) somewhat "bad" S-boxes and have a regular periodic structure.

### 2.1 XSL-Ciphers and the Notations We Use

By definition, an XSL-cipher is a composition of  $N_r$  similar rounds:

- X Before the first round, the input is XOR-ed with the key  $K_0$ . Let  $i = 1$ .
- S Then a layer of  $B$  bijective S-boxes, on  $s$  bits each, is applied in parallel.
- L Then a linear diffusion layer is applied.
- X The result is XOR-ed with another session key  $K_i$ .
- .. If  $i = N_r$ , the final result is produced.
- Otherwise  $i$  is incremented and the process goes to the step S.

We denote the key bits used in an XSL-cipher by the variables  $K_{i\ j}$  with  $i = 0..N_r$  and  $j = 1..Bs$ . There are  $N_r + 1$  session keys,  $K_0$  is the first and  $K_{N_r}$  is the last. The number of key bits before expansion is  $H_k$ , the number of key bits after expansion is  $E_k$ , and the number of expanded key bits that are linearly independent is  $L_k$ . If we pick some subset of  $L_k$  key variables  $K_{i\ j}$  that form a basis, then we will denote by  $[K_{i\ j}]$  a linear expression of this bit  $K_{i\ j}$  as a sum of the other  $K_{k\ l}$  that are in the basis.

We call  $X_{i\ j}$  the  $j$ th bit of the input of  $i$ th round S-boxes layer, step S (taken after the previous XOR with the session key X). We denote by  $Y_{i\ j}$  the  $j$ th bit of the input of the linear part L of  $i$ th round (taken after the S-box application S). Similarly let  $Z_{i\ j}$  be the  $j$ th bit of the output of the step L (before the next key XORing step X). Consequently we will denote the plaintext by  $Z_0$  and the ciphertext by  $X_{N_r+1}$ , however these are constants, not variables. To summarize we have:

Step:	X	S	L	X	...	...	S	L	X
Values:	$Z_0$	$X_1$	$Y_1$	$Z_1$	$X_2$	...	$X_{N_r}$	$Y_{N_r}$	$Z_{N_r}$ $X_{N_r+1}$

With these notations we obtain  $X_{i+1\ j} = Z_{i\ j} \oplus K_{i\ j}$  for all  $i = 0..N_r$ .

2.2 Top-Level Structure of Rijndael

Rijndael specified in [4], is a special type of XSL-cipher with  $s = 8$  and  $B = 4N_b$ . We will not give a full description of it, but will recall all the essential facts when necessary. Rijndael has  $N_r = 10 \dots 14$  rounds. The data in Rijndael is represented as rectangular “states” that are composed of  $N_b$  columns, each having the size of 4 S-boxes ( $4s = 32$  bits). We have either  $N_b = 4, 6$  or  $8$ , which gives block sizes of  $32N_b = 128, 192$  and  $256$  bits respectively. The encryption in Rijndael is performed as follows:

- X The input sequence  $Z_{0\ j}$  is XOR-ed with the session key  $K_{0\ j}$ . Let  $i = 1$ .
- S Then resulting sequence  $X_{i\ j}$  is transformed by  $B = 4N_b$  S-boxes on  $s = 8$  bits each.
- L Then resulting sequence  $Y_{i\ j}$  is then subject to a composition of two linear transformations. First we have a permutation of bytes called ShiftRow, then four linear transformations MixColumn:  $GF(256)^4 \rightarrow GF(256)^4$  applied in parallel for each of  $N_b$  columns.  
If  $i = N_r$  (the last round) we have only ShiftRow, the MixColumn is omitted.
- X Then resulting sequence  $Z_{i\ j}$  is XOR-ed with another session key  $K_i$  producing, either the ciphertext (if  $= N_r$ ), or the process increments  $i$  and goes to step S.

The (unexpanded) key length is  $H_k = 32N_k$  bits with  $N_k = 4, 6$  or  $8$ , thus again  $128, 192$  and  $256$  bits respectively. It is then expanded to  $E_k = (N_r + 1)Bs = (N_r + 1)N_b \cdot 32$  bits.

2.3 Top-Level Structure of Serpent

Serpent described in [1] is an XSL-cipher with  $s = 4$ ,  $B = 32$  and  $N_r = 32$ . The block size is always  $128$  bits. The key length can be  $H_k = 128, 192$  or  $256$  bits, and is also expanded to  $E_k = (N_r + 1)Bs = 1056$  bits.

3 S-boxes and Overdefined Algebraic Equations

The only non-linear part of XSL-ciphers are the S-boxes. Let the function  $F : GF(2)^s \rightarrow GF(2)^s$  be such an S-box, given an input  $x = (x_1..x_s)$  we obtain an output  $y = (y_1..y_s) = F(x)$ . In Rijndael and Serpent, like for all other “good” block ciphers, the S-boxes are build with “good” boolean functions. Among the known criteria on cryptographically “good” boolean functions, we know that  $y_i$  should have a high algebraic degree in the  $x_i$ . However, this does not assure that there is no other “implicit” multivariate equations of the form

$P(x_1, \dots, x_s, y_1, \dots, y_s)$  that are of low algebraic degree. We will show that for Rijndael, and for Serpent, for very different reasons, a great number of such equations exist. We are interested in the actual number  $r$  of such equations  $P(x_1, \dots, x_s, y_1, \dots, y_s)$ , being of low degree  $d$ , e.g.  $d \leq 2$ . Unlike for “explicit” equations  $y_i = f(x_1, \dots, x_s)$ , this number  $r$  can be bigger than  $s$ . We are also interested in the total number of monomials  $t$  that appear in these equations, counted including the constant term. With these notations:

- In general,  $t \approx \binom{s}{d}$ . If  $t \ll \binom{s}{d}$ , we say that the equations are **sparse**.
- When  $r \approx s$ , the equations give enough information about the S-box, and the system will be sufficiently **defined** to yield about 1 solution  $x$ , given  $y = F(x)$ .

Consequently, when  $r \gg s$ , the system is said to be **overdefined**.

### 3.1 Quality of S-boxes and Random S-boxes

When  $r$  is close to  $t$ , we may eliminate most of the terms by linear elimination, and obtain simpler equations that are sparse and maybe even linear. For this reason, it is possible to measure the quality of our system of equations by the ratio  $t/r \geq 1$ . If  $t/r$  is close to 1, the S-box is considered as “bad”. From this point of view, both overdefined systems (big  $r$ ) and sparse systems (small  $t$ ) will be “bad”. Otherwise, if the system is not overdefined and not sparse,  $t/r \approx \mathcal{O}(s^{d-1})$ , and such an S-box will be “good” (unless  $s$  is very small). We will see that the actual contribution of the S-boxes to the complexity of the attacks described in this paper is approximatively  $\Gamma = ((t-r)/s)^{\lceil (t-r)/s \rceil}$ . It is possible to show that for a random S-box, the smallest value of  $\Gamma$  that can be achieved will be double-exponential in  $s$ . However it will be still relatively small for Serpent ( $s = 4$ ). For different reasons, the Serpent and Rijndael S-boxes can both be described by overdefined systems with quite a small  $\Gamma$ .

### 3.2 Overdefined Equations on the Serpent S-box

We show that 4-bit S-boxes always give an overdefined system of quadratic equations. Consider a  $16 \times 37$  matrix containing in each row the values of the  $t = 37$  monomials  $\{1, x_1, \dots, x_4, y_1, \dots, y_4, x_1x_2, \dots, x_1y_1, \dots, y_3y_4\}$  for each of the  $2^s = 16$  possible entries  $x = (x_1, \dots, x_4)$ . The rank of this matrix is at most 16, therefore whatever is the S-box, there will be at least  $r \geq 37 - 16 = 21$  quadratic equations. This is a very overdefined system since  $21 \gg 4$ . We have  $t/r \approx 1.75$  and  $\Gamma = ((t-r)/s)^{\lceil (t-r)/s \rceil} = 2^{8.0}$ . We note that a smaller  $t/r$  would be achieved with cubic equations on this S-box, but  $\Gamma$  would be much bigger then. It is also possible to consider bi-affine equations. In this case we have  $t = 25$  and  $r \geq 25 - 16 = 9$  which is still overdefined, it gives the same value of  $\Gamma = 2^{8.0}$ .

### 3.3 Overdefined Equations on the Rijndael S-box

For Rijndael we have  $s = 8$ . It is easy to see that with the method described above in Section 3.2, a random S-box on 8 bits will give  $r = 0$  because  $2^s = 256$

is bigger than the number 137 of possible quadratic terms. We see that  $s = 8$  is quite big compared to Serpent: there are  $(2^8)! \approx 2^{1684}$  bijective S-boxes on 8 bits, compared with only  $(2^4)! \approx 2^{44}$  for  $s = 4$ . We don't expect any useful properties to happen by chance. Still, the design of the Rijndael S-box induces a lot of algebraic structure, see [42]. This yields very special properties.

Rijndael S-box is a composition of the "patched" inverse in  $GF(256)$  with 0 mapped on itself, with a multivariate affine transformation  $GF(2)^8 \rightarrow GF(2)^8$ . Following [4] we call these functions  $g$  and  $f$  respectively, and denote  $S = f \circ g$ . Let  $x$  be an input value and  $y = g(x)$  the corresponding output value. We also note  $z = S(x) = f(g(x)) = f(y)$ . According to the definition of the S-box:

$$\forall x \neq 0 \quad 1 = xy$$

This equation gives, in turn, 8 multivariate bi-linear equations in 8 variables and this leads to 8 bi-affine equations between the  $x_i$  and the  $z_j$ . It is possible to see that 7 of these equations are true with probability 1, and the 8th is true with probability 255/256. The existence of these equations for  $g$  and  $S$  is obvious. Surprisingly, much more such equations exist. For example we have:

$$x = y * x^2$$

Since  $x \mapsto x^2$  is linear, if written as a set of 8 multivariate functions, the above equation gives 8 bi-affine equations between the  $x_i$  and the  $y_j$ , and, in turn, between the  $x_i$  and the  $z_j$ . Adding the fact that the above equation is symmetric with respect to the exchange of  $x$  and  $y$ , we get 16 bi-affine equations true with probability 1 between the  $x_i$  and the  $z_j$ .

From the above we have 23 quadratic equations between  $x_i$  and the  $z_j$  that are true with probability 1. We have explicitly computed these equations (see the extended version of this paper), verified that they are all linearly independent, and also that there are no more such bi-affine equations<sup>2</sup>. The number of terms present in these equations is  $t = 81$ . These terms are:  $\{1, x_1, \dots, x_8, z_1, \dots, z_8, x_1 z_1, \dots, x_8 z_8\}$ , and there is no terms  $x_i x_j$  or  $z_i z_j$ . We get  $t/r \approx 3.52$  and  $\Gamma \approx 2^{22.9}$ , much more than for Serpent.

**An Additional 24th Equation:** We observe that in Rijndael S-box, if  $x$  is always different than 0, there 24 linearly independent quadratic equations. For one S-box, the probability of this 24th equation to be true is 255/256. We are interested in probability that it is true for **all** S-boxes in the execution of Rijndael (i.e. we have  $x \neq 0$  everywhere). As it has been already pointed out by the authors of [8], this probability is quite big. It is in fact:

$$(255/256)^{4 \cdot N_b N_r + 4 \cdot \lceil \frac{N_b(N_r+1) - N_k}{N_k} \rceil + 4 \cdot 1_{N_k=8} \cdot \lceil \frac{N_b(N_r+1) - N_k - 4}{N_k} \rceil}$$

<sup>2</sup> If we square the equation  $x = x^2 * y$  we obtain successively  $x^2 = x^4 * y^2, \dots, x^{128} = x * y^{128}$ . It can be seen that each of them also gives 8 bi-affine equations. However, since the square is multivariate linear, each of them produces **the same** 8 equations, modulo a linear combination.

This gives between  $1/2$  for the smallest Rijndael 128 bits and about  $1/9$  for the biggest 256-bit version. Therefore, if an attack works better with 24 equations, and uses only one (or two) executions of the cipher it will be interesting to use  $r = 24$  and repeat the whole attack a few times. Otherwise, we use  $r = 23$ .

**Fully Quadratic Equations:** It is possible to see that if we consider fully quadratic equations, not only bi-affine, for each S-box of Rijndael there are  $r = 39$  quadratic equations with  $t = 137$ . The additional 16 equations come from the following two equations:

$$\begin{cases} x^4 y = x^3 \\ y^4 x = y^3 \end{cases}$$

However, when  $r = 39$ ,  $t = 137$ , we have  $\Gamma \approx 2^{47.0}$  instead of  $2^{22.9}$  and we always obtained worse results in our attacks, than with  $r = 23$ ,  $t = 81$ ,

**About Inverse-Based S-boxes:** In general, it is easy to see that if the S-box on  $s$  bits is an affine transformation of the inverse function in  $GF(2^s)$ , then it will give  $3s - 1$  bi-affine equations true with probability 1, and one additional equation true with probability  $1 - \frac{1}{2^s}$ . We conjecture for all  $s$  there are no more such equations (we verified this for several  $s$ ). Up till now, it seemed a very good idea to use such S-boxes: the inverse function (and its affine equivalents) has meaningful **optimality** results with regard to linear, differential and high-order differential attacks, see [216]. However in our computer simulations, done for many permutations including all the possible powers in  $GF(2^s)$ , the inverse (and its equivalents) was always the **worse** in terms of the number of such bi-affine equations. It is an open problem to find any other non-linear function  $GF(2^s) \rightarrow GF(2^s)$  that admits so many equations, for some  $s > 0$ . Therefore, we do not advocate to use such S-boxes even if they are probably still very secure.

**Related Work:** The equations we have found for the Rijndael S-box are exactly of the same type and of very similar origin, as the equations that Jacques Patarin have discovered in 1988 for the Matsumoto-Imai cryptosystem [17]. The existence of such equations for Rijndael S-boxes have been first discovered (but not published) by Courtois, Goubin and Patarin, as soon as Rijndael have been proposed as AES in 2000. Recently, in [14], Murphy and Robshaw pointed out that it is more interesting to manipulate equations over  $GF(256)$ . It leads to systems that are identical (or very similar) in terms of the number of equations and number of variables involved. However, the number of different monomials  $t$  present is lower, which is expected to give better results for our attacks.

## 4 MQ Attacks on Block Ciphers

Given an SA-cipher with S-boxes that can be described in terms of some algebraic equations, recovering the key can be written as a problem of solving a system of

such equations. If these equations are multivariate quadratic, we call this “the MQ attack”. Such equations exist for Rijndael and Serpent, as shown above in Sections 3.3 and 3.2 respectively.

#### 4.1 Attack Scenarios

There are many ways in which the MQ attack can be applied. The system of equations should be constructed in such a way that it has exactly one solution. A system that has one solution on average is sufficient in practice, and if there are a few solutions, prior to the solving stage, we would guess and fix a few bits.

**First (General) Attack Ignoring the Key Schedule.** This attack is designed for any XSL-cipher, whatever is the key schedule. For simplification we only consider the known plaintext attack. There are  $(N_r + 1)$  keys  $K_i$  that are of the same size as a plaintext, and we need enough equations to determine them uniquely. Hence we need  $(N_r + 1)$  known plaintexts. This attack scenario will be used in Section 6.

**Second (Specific) Attack Using the Key Schedule.** This attack is less general and relies on the fact that the key schedules in Rijndael and Serpent are very similar to the ciphers themselves: they use a combination of affine transformations and (the same) S-boxes. Due to the lack of space, we only study the first (more general) scenario and the second will be studied in a separate paper.

**Stronger Attack Scenarios.** If attacks based on MQ are possible, i.e. there are efficient methods to solve quadratic equations, then they allow to attack block ciphers in very strong scenarios. For example ciphertext-only attacks will be possible if the attacker is able to characterize the redundancy of the plaintext in terms of quadratic equations.

#### 4.2 Direct MQ Attack on Rijndael and Serpent:

It can be seen that in the second attack scenario, the problem of recovering the key of the 128-bit Rijndael, amounts to solving a system of 8000 quadratic equations with 1600 variables. See Appendix A for details. Similarly, the 128-bit Serpent would give a system of  $(N_r + 1)Br + N_rBr = 43680$  equations with  $(N_r + 1)Bs + (N_r - 1)Bs = 8192$  variables.

In the remaining part of the paper we study solving such (and similar) systems of equations. Our results are given in Sections 5.2 and 7.

## 5 Generic Methods for Solving Multivariate Quadratic Equations

MQ is known to be an NP-hard problem [23]. Several public key cryptosystems are based on MQ, for example HFE [19]. However, little is known about the actual hardness of MQ in practice. From the above it is clear that if this problem was very easy for 1600 variables, then Rijndael would be broken. With current attacks, factoring a 1600-bit RSA modulus provides a security level slightly lower than  $2^{128}$  [24]. Therefore, MQ should be at least as hard as factoring.

### 5.1 Solving MQ with the XL Algorithm

In [22] Shamir and Kipnis made an important discovery about the MQ problem: solving it should be much easier for overdefined systems<sup>3</sup>. This idea has been developed and consolidated in [23]. An algorithm called XL is developed for this problem. It seems that for a random system of quadratic equations over  $GF(2)$  (or one that looks random) that has a unique solution, the XL method should always work (but maybe not for some very special systems). In [13] T.T. Moh states that “From the theory of Hilbert-Serre, we may deduce that the XL program will work for many interesting cases for  $D$  large enough”. From [23] it appears that XL would be polynomial for very overdefined systems, and it seems that a variant of XL might even be subexponential in general (not only for overdefined systems). However, very little is known about the actual behaviour of XL for very big systems of equations and one can only talk about conjectured complexities.

### 5.2 First Attempt to Cryptanalyse Rijndael with XL

For the 128-bit Rijndael with 128-bit key, following the Theorem A.3.1 we get a system of  $m = 8000$  equations with  $n = 1600$  variables. Following the complexity evaluation of XL from [23], the complexity would be about  $\binom{n}{n/\sqrt{m}}^\omega \approx 2^{330}$ , assuming  $\omega = 2.376$ , the best known Gaussian reduction exponent, see [3].

This attack fails because for a random system of quadratic  $R_{ini} = m = 8000$  equations with  $n = 1600$  variables, we have about  $T_{ini} \approx n^2/2 \approx 2^{20}$  terms. This gives  $R_{ini}/T_{ini} \approx 2^{-7.3}$  that is very small and the XL algorithm has to do extensive work in order to achieve an expanded system with  $R/T \approx 1$ . It is easy to see that in our whole system  $T_{ini} \approx (8 \cdot 32 + 8 \cdot 32 + 8 + 32 + 8)(N_r \cdot 4 \cdot N_b)$  and this gives only  $R_{ini}/T_{ini} \approx 2^{-3.5}$ . Therefore there **should** be a better attack. In the next Section 6.2 we will write the quadratic equations in a different way in order to achieve an even higher value of  $R_{ini}/T_{ini}$ .

## 6 XSL Attack on Block Ciphers

In this section we will write a system of equations that describe uniquely the secret key of the cipher, following the first attack scenario from Section 4.1 that

<sup>3</sup> In this paper we will show that if the MQ is sparse, it is even much easier to solve.

does not depend on the key schedule. In order to solve these equations, we are going to introduce an improved version of the XL approach from [23], that takes advantage of their specific structure and sparsity. We call it “the XSL algorithm” where XSL stands for: “eXtended Sparse Linearization” or “multiply(**X**) by Selected monomials and **L**inearize”. In the XL algorithm, we would multiply each of the equations by all possible monomials of some degree  $D - 2$ , see [23]. Instead we will only multiply them by carefully selected monomials. It seems that the best thing to do, is to use products of monomials, that already appear in other equations.

## 6.1 Final Step and Working Condition of the XSL Attacks

In [23], when  $R \geq T$ , we have as many equations as the number of terms that appear in these equations and the big system is expected to be solved by adding a new variable for each term, and solving a linear system (doing this is known as linearization). There is no need to have  $R$  much bigger than  $T$ . because obviously, the number of linearly independent equations (denoted later by  $Free$ , cannot exceed  $T$ . In the original paper about XL [23], the system was solved when  $T - Free$  was a small number. Still it is easy to see that both XL and XSL algorithms can be extended to the case when  $T - Free$  is very big (!).

Let  $x_1$  be a variable, and let  $T'$  be the number of terms that can be multiplied by  $x_1$  and still belong to the set of  $T$  terms. Now we assume that  $Free \geq T - T' + C$  with a small  $C$ . We apply the following algorithm called “the  $T'$  method”.

1. By one single gaussian elimination we bring the system to a form in which each term is a known linear combination of the terms in  $T'$ .
2. We do the same pre-computation two times, for example with  $T'$  defined for  $x_1$  and separately for  $x_2$ .
3. In each of the two systems, we have a subsystem of  $C$  equations that contain only terms of  $T'$ . These new equations are probably **not** of the same kind that the initial equations generated in XL-like attacks: only combining all the equations one can obtain some information about the solution.
4. In each of the two subsystems of exceeding  $C$  equations, we multiply each equation by  $x_1$  and  $x_2$ , respectively. Then we substitute the expressions from point 1 in these to get some **other** equations that contain only terms of  $T'$ , but for the other variable. These equations are expected to be new and different. First because the equations from point 2 are believed to contain “some information” about the solution, and moreover if we are over  $GF(2)$  we will interact with the equation of the field  $GF(2)$  that is not necessarily done elsewhere. We have done some computer simulations that show that this heuristic works very well. See also Appendix B for an example.
5. Thus, if at the beginning  $Free \geq C + T - T'$  we can “grow” the number of equations. For now we expect to up to  $2C$  additional equations.
6. We expect that the number of new equations grows at exponential rate<sup>4</sup>.

<sup>4</sup> However, even if it grows by 1 each time, the attack will work as predicted.



7. If the initial system had a unique solution, we expect that we will end up with  $Free = T$  or  $Free = T - 1$ .
8. For each equation containing only terms in  $T'$ , the cost to compute a derived additional equation will be about  $T'^2$ . Since there are  $T'$  equations missing, we expect to do about  $T'^3$  additional operations in the attack, which can probably be reduced to  $T'^\omega$  and thus will be smaller than  $T^\omega$ .
9. If the whole attack fails one should try with another couple of variables instead of  $x_1$  and  $x_2$ , or use three variables from the start (and three systems). We conjecture that three variables should always be sufficient. The number of possibilities grows very fast with the number of variables, a new equation obtained with one variable can be immediately transformed and expanded with **all** the other variables.

For example, in our attack on Rijndael 256 bits given in Section 7.1 we have  $T \approx 2^{125}$  and  $T' \approx 2^{114}$ . The attack is expected to work as long as  $Free > T - T'$ .

## 6.2 Core of the XSL Attacks

In this version of the XSL attack we assume that the system of equations for each S-box is overdefined and  $r \geq 2s$ . Let  $S$  be the total number of S-boxes in our attack. Since we are going to use the most general attack scenario described in 4.1 that ignores the key schedule of the cipher, we will have to consider  $N_r + 1$  executions of the cipher, see Section 4.1.  $S$  will be equal to

$$S = B \cdot N_r(N_r + 1).$$

### Equations on the S-boxes and Their Multiples

Let  $A$  be an S-box of a XSL-cipher, called “active S-box”. We write:

$$0 = \sum \alpha_{ijk} X_{i \ j} Y_{i \ k} + \sum \beta_{ij} X_{i \ j} + \sum \gamma_{ij} Y_{i \ j} + \delta.$$

The total number of terms (i.e. all monomials including the constant) that appear in these equations is small, only  $t$  (most of them of the form  $X_{i \ j} Y_{i \ k}$ ). For this reason (unlike in Appendix A) we use both the original variables  $X_{i \ j}$  and  $Y_{i \ k}$ .

We will not use these equations directly, but we will, from these equations, and separately for each S-box, choose some  $t - r$  terms as a basis, and write the expression of each of the remaining  $r$  terms as a linear combination of the  $(r - t)$  terms for the same S-box. We will choose a basis such that all the terms  $X_{i \ j}$  and  $Y_{i \ j}$  are not in the basis and such that 1 is in the basis. This is possible because  $r \geq 2s$ . Each time, in the attack we want to use one of the other  $r$  terms, we will directly write them as the linear combination of the elements of the basis. We define  $[X_{i \ j}]$  and  $[Y_{i \ j}]$  as precisely these linear combinations of the  $(t - r)$  elements of the basis.

**Note:** This can be called “a compact version of the first XSL attack.” A different approach is possible that uses all the  $t$  terms for each S-box (and later their products). This gives different results and will be studied in a separate paper.

## Products of Terms

The critical parameter of our attack will be  $P \in \mathbb{N}$ . We will manipulate products of up to  $P$  terms that come from  $P$  different S-boxes. The total number of terms used in the attack is about:

$$T \approx (t - r)^P \cdot \binom{S}{P}$$

Moreover, we have (see the definition of  $T'$  given in Section 6.1 above):

$$T' \approx t'(t - r)^{P-1} \cdot \binom{S-1}{P-1}$$

with  $t' < t$  being the number of terms in the basis for one S-box, that can be multiplied by some fixed variable  $X_{i-j}$ , and are still in the basis. For example for Rijndael we use  $r = 23$ ,  $t = 81$ , and get  $t' = 9$ .

### 6.3 Equations on the Diffusion Layers

We will construct a set of equations in such a way that they can be multiplied by many products of terms, and that all the resulting product can be written using only the products of up to  $P$  terms, that are taken in the respective bases we have chosen for  $P$  different S-boxes. We will get equations that are linear combinations of the  $T$  monomials, as defined above. It seems that the best way to attack our problem is to completely eliminate all the key variables and write all possible equations of the form:

$$[X_{i-j}] \oplus \sum \alpha_j [Y_{i-1-j}] = [X'_{i-j}] \oplus \sum \alpha_j [Y'_{i-1-j}] = [X''_{i-j}] \oplus \sum \alpha_j [Y''_{i-1-j}] = \dots$$

The expressions  $[X_{i-j}]$  and  $[Y_{i-j}]$  have been defined above, they are linear combinations of quadratic terms that are the elements of the basis.

We have  $N_r(N_r + 1)(Bs)$  such equations. Each of these equations, called “active equation”, will be multiplied by products of terms for some  $(P - 1)$  “passive” S-boxes. Here we need to exclude the terms for a few neighbouring S-boxes (i.e. that have common variables with the active equation), it does not change a lot the number of equations generated. The number of new equations is called  $R$ . It is approximatively:

$$R \approx N_r(N_r + 1)(Bs) \cdot (t - r)^{P-1} \cdot \binom{S}{P-1} \approx S \cdot s (t - r)^{P-1} \cdot \binom{S}{P-1}$$

### 6.4 Expected Complexity of the XSL Attack

The goal of the attack is to obtain  $R > T - T'$ . It gives:

$$Ss(t - r)^{P-1} \binom{S}{P-1} > (t - r)^P \binom{S}{P} - t'(t - r)^{P-1} \binom{S-1}{P-1}$$

$$\frac{S^2 s}{S - P + 1} > \frac{(t - r)S}{P} - t'$$

We will assume that  $P \ll S$  ( $S$  is usually quite big) and thus  $S - P + 1 \approx S$ .

$$s > \frac{(t - r)}{P} - \frac{t'}{S}$$

$$s + \frac{t'}{S} > \frac{(t - r)}{P}$$

We see that this condition can always be satisfied, if  $P$  is sufficiently big. We get that:

$$P \geq \frac{(t - r)}{s + \frac{t'}{S}} \quad (\#)$$

**Note:** From this it might seem that the XSL attack will work for  $r = 0$ , however we have previously assumed that  $r \geq 2s$ . Therefore  $r = 0$  is not possible.

Let  $T^\omega$ , be the complexity of the Gaussian reduction, the complexity of the attack is about:

$$WF = T^\omega \approx (t-r)^{\omega P} \left(\frac{S}{P}\right)^\omega \approx (t-r)^{\omega P} (B \cdot N_r^2)^{\omega P} \approx \left(\frac{t-r}{s} \cdot Bs \cdot N_r^2\right)^{\omega P}$$

Now let us apply the estimation (#). In practice the value  $\frac{t'}{S}$  will be very small, and vanishes for big ciphers (big  $N_r$  or big  $B$ ). Therefore we assume that  $P = \lceil \frac{t-r}{s} \rceil + o(1)$ . It gives the following (rough) estimation of the complexity of the XSL attack on block ciphers, again assuming that  $r \geq 2s$ :

$$WF \approx \left(\frac{t-r}{s}\right)^{\omega \lceil \frac{t-r}{s} \rceil + o(1)} \cdot (Bs \cdot N_r^2)^{\omega \lceil \frac{t-r}{s} \rceil + o(1)}$$

$$WF = \Gamma^\omega \cdot (\text{Block size})^{\omega \frac{t-r}{s}} (\text{Number of rounds})^{2\omega \frac{t-r}{s}}$$

This is polynomial in the block size and the number of rounds. The constant part depends on  $\Gamma$  that depends only on the parameters of the S-box used in the cipher, and is in general double-exponential in  $s$ , see Section 3.1. For a given cipher the constant part  $\Gamma^\omega$  in the complexity of XSL will be fixed (but usually very, very big).

## 6.5 Actual Complexity of the XSL Attacks

In the above derivation we assumed that all the equations in  $R$  are linearly independent and this implies that for some fixed  $P$  the attack will always work for any number of rounds. From our early simulations<sup>5</sup> it seems that the attack works for many rounds and then it fails. Thus  $P$  would rather increase (but slowly) with the number of rounds.

<sup>5</sup> See, the second XSL attack, preliminary version, .  
<http://eprint.iacr.org/2002/044/>

If  $P$  were constant, for a fixed S-box that have many overdefined equations, the XSL attack will be polynomial in the number of rounds. Even if  $P$  grows slowly, and XSL is subexponential, it would be already an important breakthrough, as the complexity of the classical attacks on block ciphers, such as linear or differential cryptanalysis, grows exponentially in the number of rounds (and so does the number of required plaintexts).

In fact there is another way to see that there is a risk that the problem to break Rijndael might be subexponential when the number of rounds grows. Indeed, in this paper we show how to write Rijndael as an overdefined system of quadratic equations, with size that is linear in  $N_r$ , see Appendix A. The problem of solving such a system of quadratic equations over  $GF(2)$  might already be subexponential using the original XL algorithm from [23]. See Section A.4 for more comments on this. Finally, our equations from Appendix A are also overdefined, sparse and have a lot of structure, which also should help the attacker.

## 7 Consequences of the XSL Attack

### 7.1 Application to Rijndael

We consider the 128-bit Rijndael with 256 bit keys. We have  $N_b = 4, N_k = 8, N_r = 14, s = 8, r = 23, t = 81, t' = 9, S = 3360$ , then for  $P = (t - r)/(s + t'/S) = 8$ , computed following (#), we get  $T \approx 2^{125}, T' \approx 2^{114}, R \approx 2^{125}$  with  $\frac{R}{T-T'} = 1.106$ . The result is:

$$T^\omega \approx 2^{298}$$

This version of the XSL attack fails also for other variants of AES. We expect that much better results should be obtained with the combination of the second XSL attack [5], with equations over  $GF(256)$  as proposed by Murphy and Robshaw [14]. It is not excluded that even AES-128 could be broken: for  $N_b = 4, N_k = 4, N_r = 10, s = 8, r = 24, t = 41, t' = 4, S = 201$ , our early estimation gives that for  $P = 3$  we have  $T \approx 2^{36}, T' \approx 2^{27}, R \approx 2^{36}, R' \approx 2^{33}$ , and  $\frac{R+R'}{T-T'} = 1.01$ . If this attack worked as well as expected [5], the resulting complexity would be  $T^\omega \approx 2^{87}$ .

### 7.2 Application to Serpent

For 256-bit Serpent, we have  $N_r = 32, s = 4, r = 21, t = 37, t' = 5, S = 33792$ . Then for  $P = (t - r)/(s + t'/S) = 5$ , we get  $T \approx 2^{88}, T' = 2^{74}, R = 2^{88}, \frac{R}{T-T'} = 1.25$ . The result is:

$$T^\omega \approx 2^{210}$$

It seems that the XSL attack breaks 256 bit Serpent. Though it is obtained with the fairly theoretical  $\omega = 2.376$  from [3], using Strassen's exponent we still get  $2^{245}$ . It is however not proven that the attack will work as predicted for  $P = 5$ . Though XSL attacks will probably always work for some  $P$ , we considered the minimum value  $P$  for which  $\frac{R}{T-T'} \geq 1$ . This condition is necessary, but probably not sufficient. A small change (e.g. increase by 1 or 2) in  $P$  would lead to a dramatic overload in the complexity, going beyond the exhaustive search.

### 7.3 Consequences for the Design of Block Ciphers

There are two complementary approaches in the block cipher design that could be seen in the AES contest. Either a cipher is designed with a very small number of rounds that are very complex (for example in DFC), or it has a large number of rounds that are very simple (for example in Serpent). In [27] the authors warn that: “an attack against Serpent may hold for any set of (random) S-boxes”. It seems that we have found such an attack and using many layers of very simple S-boxes is maybe not such a very good idea. Still, a correct choice of parameters will prevent the attacks.

For different reasons, the XSL attack is also applicable to all ciphers in which the only non-linear part is the inverse function in  $GF(2^s)$ , with a small  $s$ . Therefore, ciphers such as Rijndael and Camellia should either use  $s$  that is sufficiently large, maybe  $s > 8$ , or consider different S-boxes. This last possibility should give new optimal designs of S-boxes, not only close to optimal in terms of linear and differential attacks, but also incorporating our new criterion, i.e. having a big value of  $\Gamma$ , for example  $\Gamma > 2^{32}$ .

Even if the attacks of the present paper have not yet been tested on really big examples, they are an important threat for ciphers such as Rijndael, Serpent and Camellia. We propose that all block ciphers should apply the following criterion (due originally to Shannon [25]): The attacker should not be able to write a system of algebraic equations of simple type and of reasonable size, that completely characterizes the secret key. It can be achieved if one uses at least a few (relatively) big randomly generated S-boxes.

## 8 Conclusion

In this paper we point out an unexpected property of Rijndael and Serpent: they can be described as a system of overdefined and sparse quadratic equations over  $GF(2)$ . It was known from [23] that solving such systems is easier if they are overdefined, and the problem might even be subexponential (conjectured) for small fields such as  $GF(2)$ . It is therefore possible that the security of Rijndael and Serpent would not grow exponentially with the number of rounds.

A direct application of the XL attack from Eurocrypt’00 is extremely inefficient. Knowing that the equations are not only overdefined, but also sparse and structured, we have introduced a new method called XSL. If the XSL attack works as well predicted, it might (marginally) break Serpent 256 bits. With equations over  $GF(2)$  we do not get an efficient attack for AES. However a different version of XSL combined with equations over  $GF(256)$  is expected to give much better results. In order to prevent such attacks, we propose that at least a few S-boxes in a cipher should not be described by a small system of overdefined multivariate equations.

## Acknowledgments

The following people have contributed to this paper: Mehdi-Laurent Akkar, Don Coppersmith, Louis Goubin, Philip Hawkes, Jacques Patarin, Pham Thi Minh Tam, and the anonymous reviewers of Asiacypt.

## References

1. Ross Anderson, Eli Biham and Lars Knudsen: *Serpent: A Proposal for the Advanced Encryption Standard*. Available from <http://www.cl.cam.ac.uk/~rja14/serpent.html>
2. Anne Canteaut, Marion Videau: *Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis*; Eurocrypt 2002, LNCS 2332, Springer.
3. Don Coppersmith, Shmuel Winograd: "Matrix multiplication via arithmetic progressions"; J. Symbolic Computation (1990), 9, pp. 251-280.
4. Joan Daemen, Vincent Rijmen: *AES proposal: Rijndael*; The latest revised version of the proposal is available on the internet, <http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf>
5. Nicolas Courtois, Louis Goubin, Willi Meier, Jean-Daniel Tacier: *Solving Under-defined Systems of Multivariate Quadratic Equations*; PKC 2002, LNCS 2254, Springer, pp. 211-225.
6. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*; Cryptographers' Track Rsa Conference 2001, San Francisco 8-12 April 2001, LNCS2020, Springer-Verlag, pp. 266-281.
7. Horst Feistel: *Cryptography and computer privacy*; Scientific American, vol. 228, No. 5, pp. 15-23, May 1973.
8. Niels Ferguson, Richard Schroeppel and Doug Whiting: *A simple algebraic representation of Rijndael*; SAC'01, page 103, LNCS 2259, Springer.
9. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting: *Improved Cryptanalysis of Rijndael*, FSE 2000, Springer.
10. J.B. Kam and G.I. Davida: *Structured design of substitution-permutation encryption networks*; IEEE Trans. on Computers, Vol. C-28, 1979, pp.747-753.
11. Lars R. Knudsen, Vincent Rijmen: *On the Decorrelated Fast Cipher (DFC) and its Theory*; FSE'99, Springer, LNCS 1636, pp. 81-94.
12. Michael Luby, Charles W. Rackoff, *How to construct pseudorandom permutations from pseudorandom functions*; , SIAM Journal on Computing, vol. 17, n. 2, pp. 373-386, April 1988.
13. T.T. Moh: *On The Method of XL and Its Inefficiency Against TTM*, available at <http://eprint.iacr.org/2001/047/>.
14. S. Murphy, M. Robshaw: *Essential Algebraic Structure within the AES*, Crypto 2002, Springer.
15. Moni Naor and Omer Reingold: *On the construction of pseudo-random permutations: Luby-Rackoff revisited*; Journal of Cryptology, vol 12, 1999, pp. 29-66.
16. Kaisa Nyberg: *Differentially Uniform Mappings for Cryptography*; Eurocrypt'93, LNCS 765, Springer, pp. 55-64.
17. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*; Crypto'95, Springer-Verlag, pp. 248-261.
18. Jacques Patarin: *Generic Attacks on Feistel Schemes*; Asiacypt 2001, LNCS 2248, Springer, pp. 222-238.

19. Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms*; in Eurocrypt'96, Springer Verlag, pp. 33-48.
20. Jacques Patarin, Nicolas Courtois, Louis Goubin: *Improved Algorithms for Isomorphism of Polynomials*; Eurocrypt 1998, Springer-Verlag.
21. Adi Shamir, Alex Biryukov: *Structural Cryptanalysis of SASAS*; Eurocrypt 2001, LNCS 2045, Springer, pp. 394-405.
22. Adi Shamir, Aviad Kipnis: *Cryptanalysis of the HFE Public Key Cryptosystem*; In Advances in Cryptology, Proceedings of Crypto'99, Springer-Verlag, LNCS.
23. Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
24. Robert D. Silverman: *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths*; RSA Lab. report, [www.rsasecurity.com/rsalabs/bulletins/bulletin13.html](http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html).
25. Claude Elwood Shannon: *Communication theory of secrecy systems*; , Bell System Technical Journal 28 (1949), see in particular page 704.
26. Serge Vaudenay: *Provable Security for Block Ciphers by Decorrelation*; Technical Report LIENS-98-8, ENS, France, available at <http://lasecwww.epfl.ch/query.msql?ref=Vau98b>.
27. Serge Vaudenay, Shihō Moriai: *On the Pseudorandomness of Top-Level Schemes of Block Ciphers*; Asiacrypt 2000, LNCS 1976, Springer, pp. 289-302.

## A Direct MQ Attack on Rijndael

It is interesting to know how to describe Rijndael as a system of quadratic equations with a minimum number of variables and maximum number of equations. We are in the second attack scenario with one or a few known plaintexts (see Section 4.1).

### A.1 Minimizing the Number of Variables for Rijndael

For each round  $i$ , we know that there are  $4r \cdot N_b$  quadratic equations between the  $(Z_{i-1\ j} + K_{i-1\ j})$  and the  $(Z_{i\ k})$ . They are of the following form:

$$0 = \sum \alpha_{ijk} Z_{i-1\ j} Z_{i\ k} + \sum \alpha_{ijk} [K_{i-1\ j}] Z_{i\ k} + \sum \beta_{ij} Z_{i\ j} + \sum \beta_{ij} [K_{i\ j}] + \gamma.$$

Exception is made for the first round, for which the  $Z_0$  being known, they are of the form:

$$0 = \sum \alpha_{ij} [K_{0\ i}] Z_{1\ j} + \sum \beta_i Z_{1\ i} + \sum \gamma_i [K_{0\ i}] + \delta.$$

Finally, for the last round, the  $X_{N_r\ k}$  will be expressed as a sum of the known ciphertext  $Z_{N_r+1\ k}$  and  $[K_{N_r\ k}]$ , giving the equations of the form:

$$\begin{aligned} 0 = \sum \alpha_{ij} Z_{N_r-1\ i} [K_{N_r\ j}] + \sum \alpha_{ij} [K_{N_r-1\ i}] [K_{N_r\ j}] + \sum \beta_i Z_{N_r-1\ i} + \\ + \sum \beta_i [K_{N_r-1\ i}] + \sum \gamma_i [K_{N_r\ i}] + \delta. \end{aligned}$$

In all we will get  $4 \cdot r \cdot N_r \cdot N_b$  quadratic equations over  $GF(2)$ . The number of variables  $Z_{i\ j}$  is only  $4s \cdot (N_r - 1)N_b$ .

## A.2 Using the Key Schedule

We have:

$$X_{i+1\ j} = Z_{i\ j} \oplus [K_{i\ j}] \quad \text{for all } i = 0..N_r. \quad (1)$$

In order to define what are the  $[K_{i\ j}]$  we need to choose a basis for the  $K_{i\ j}$ . From the key schedule [4] it is obvious that one may take as “true key variables” all the  $N_k$  variables from the first round, then all the first columns of each consecutive key states, and if  $N_k = 8$ , also the 5th columns. By inspection we see that the number of “true key variables” is:

$$L_k = \begin{cases} 32 \cdot (N_k + \lceil (N_r \cdot N_b + N_b - N_k) / N_k \rceil) & \text{if } N_k \neq 8 \\ 32 \cdot (N_k + \lceil (N_r \cdot N_b + N_b - N_k) / 4 \rceil) & \text{if } N_k = 8 \end{cases}$$

For example, for 128-bit Rijndael with  $H_k = 128$  we have  $L_k = 32 \cdot (4 + 10) = 448$ .

**Additional Equations.** We call “redundant true variables” all the  $L_k - H_k$  additional variables that are determined by some initial subset of  $H_k$  unexpanded variables. From the key schedule we see that for each of these  $L_k - H_k$  “redundant true variables” we may write  $r = 23$  (or 24) quadratic equations. Each of the “redundant true” key state columns is a XOR of one the previous columns, a parallel application of 4 S-boxes to another column, and of a constant. Thus these equations are of the form:

$$\sum \alpha_{ijkl} [K_{i\ j}] [K_{k\ l}] + \sum \beta_{ij} [K_{i\ j}] + \gamma. \quad (2)$$

The number of these equations is:

$$r \cdot \frac{L_k - H_k}{s}$$

## A.3 Summary of the Equations and Concrete Applications

**Theorem A.3.1 (Reduction Rijndael  $\rightarrow$  MQ).** The problem of recovering the secret key of Rijndael given about one pair plaintext/ciphertext can be written as an overdefined system of

$$m = 4 \cdot r \cdot N_b \cdot N_r + r(L_k - H_k)/s$$

sparse quadratic equations with the number of unknowns being:

$$n = 4 \cdot s \cdot (N_r - 1)N_b + L_k.$$

**Concrete Application to Rijndael:** We will use fully quadratic equations obtained in Section 2. We have  $r = 39$  and  $t = 137$ , however since this attack will only require 1 or 2 known plaintexts, we may assume  $r = 40$  (see Section 2).

- Thus for the 128-bit Rijndael with 128-bit key, we can write the problem of recovering the key as a system of 8000 quadratic equations with 1600 variables.



- For the 256-bit Rijndael with 256-bit key, we get a system of 22400 quadratic equations with 4480 variables.

In general, no efficient algorithms are known to solve such big systems of equations. In fact however, they are sparse and have regular structure, see Section 5.2. In Section 6.2 we write quadratic equations in a different way, more suitable for our the XSL attacks.

#### A.4 Theoretical Consequences for Rijndael and AES

The above reduction has already some very important consequences for Rijndael and AES. We consider the security of some generalized version of Rijndael in which the number of rounds  $N_r$  increases and all the other parameters are fixed.

On one hand, in all general attacks previously known against such ciphers, for example in linear or differential attacks, the security grows exponentially with  $N_r$ . There are also combinatorial attacks such as square attack, but these will simply not work if  $N_r$  is sufficiently large. On the other hand, we observe that the number of variables (and the number of equations) in the reduction is **linear** in the number of rounds  $N_r$ . Therefore, if the MQ problem is subexponential, which seems possible from the XL paper [23], to break Rijndael would also be subexponential<sup>6</sup>, i.e. the security would **not** grow exponentially with the number of rounds  $N_r$ .

**Remark 1:** It is important to see that the result would not be the same if the reduction were for example quadratic in  $N_r$ . In this case XL could be subexponential, for example in  $n^{\sqrt{n}}$  but the Rijndael could still be fully exponential, for example in  $(N_r^2)^{N_r}$ .

**Remark 2:** It seems that the same remark will hold for any block cipher composed with rounds of fixed type: obviously each of them can always be written as a set of quadratic equations. However, in this case, the size of the system (even for one round) will be so huge that there will be no hope for any practical attacks.

## B A Toy Example for the “ $T'$ Method”

This is a concrete working example for the final step of the XSL algorithm called the “ $T'$  method”. It can also be applied to the XL algorithm.

We have  $n = 5$  variables, and thus  $T = 16$  and  $T' = 10$ . We start with a random system that has exactly one solution, and with  $Free > T - T'$  and with 2 exceeding equations, i.e.  $Free = T - T' + 2$ . Here is a system in which  $T'$  is defined with respect to  $x_1$ :

---

<sup>6</sup> It also possible that XL is subexponential only on average, and AES gives very special systems.

$$\begin{cases} x_3x_2 = x_1x_3 + x_2 \\ x_3x_4 = x_1x_4 + x_1x_5 + x_5 \\ x_3x_5 = x_1x_5 + x_4 + 1 \\ x_2x_4 = x_1x_3 + x_1x_5 + 1 \\ x_2x_5 = x_1x_3 + x_1x_2 + x_3 + x_4 \\ x_4x_5 = x_1x_2 + x_1x_5 + x_2 + 1 \\ 0 = x_1x_3 + x_1x_4 + x_1 + x_5 \\ 1 = x_1x_4 + x_1x_5 + x_1 + x_5 \end{cases}$$

Here is the same system in which  $T'$  is defined with respect to  $x_2$ :

$$\begin{cases} x_1x_3 = x_3x_2 + x_2 \\ x_1x_4 = x_3x_2 + x_2 + x_1 + x_5 \\ x_1x_5 = x_2x_4 + x_3x_2 + x_2 + 1 \\ x_3x_5 = x_2x_4 + x_3x_2 + x_2 + 1 + x_4 + 1 \\ x_3x_4 = x_2x_4 + x_1 + 1 \\ x_4x_5 = x_1x_2 + x_2x_4 + x_3x_2 \\ 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \end{cases}$$

We have  $rank = 8$ . Now multiply the two exceeding equations of the first version of the system by  $x_1$ .

$$\begin{cases} 0 = x_1x_3 + x_1x_4 + x_1 + x_1x_5 \\ 0 = x_1x_4 \end{cases}$$

We have  $rank = 10$ . We get two new linearly independent equations.

We rewrite these equations, using the second system, only with terms that can be multiplied by  $x_2$ . Now we have 4 exceeding equations for the second system (two old and two new):

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5 + x_2 + 1 \\ 0 = x_3x_2 + x_2 + x_1 + x_5 \end{cases}$$

We multiply these four equations by  $x_2$ .

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_2x_4 + x_2 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5x_2 \\ 0 = x_3x_2 + x_2 + x_1x_2 + x_2x_5 \end{cases}$$

We are not lucky, the second equation is invariant by this transformation. Still we get three new linearly independent equations. We have  $rank = 13$ .

We rewrite, using the first system, the three new equations with terms that can be multiplied by  $x_1$ .

$$\begin{cases} 1 = x_1x_5 + x_2 + x_3 + x_4 \\ 1 = x_1x_2 + x_1x_3 + x_1x_5 + x_2 + x_3 + x_4 \\ 0 = x_3 + x_4 \end{cases}$$

Still  $rank = 13$ . Then we multiply the three new equations by  $x_1$ :

$$\begin{cases} 1 = x_1x_5 + x_1x_2 + x_1x_3 + x_1x_4 \\ 1 = x_1x_5 + x_1x_4 \\ 0 = x_3 + x_4 \end{cases}$$

We have  $rank = 14$ . We get one more linearly independent equation. The two other are redundant. Now we rewrite the first equation with terms that can be multiplied by  $x_2$ :

$$0 = x_1x_2 + x_2x_4 + x_3x_2 + x_1 + x_2 + x_5$$

We have still  $rank = 14$ . Then we multiply the new equation by  $x_2$ :

$$0 = x_2x_4 + x_3x_2 + x_2x_5 + x_2$$

We get another new linearly independent equation. We have  $rank = 15$ . The rank is the maximum that can be achieved, there are 15 non-zero monomials here, and  $rank = 16$  can only be achieved for a system that is contradictory.

# Analysis of Neural Cryptography

Alexander Klimov, Anton Mityagin, and Adi Shamir

Computer Science Department, The Weizmann Institute, Rehovot 76100, Israel,  
`{ask,mityagin,shamir}@wisdom.weizmann.ac.il`

**Abstract.** In this paper we analyse the security of a new key exchange protocol proposed in [3], which is based on mutually learning neural networks. This is a new potential source for public key cryptographic schemes which are not based on number theoretic functions, and have small time and memory complexities. In the first part of the paper we analyse the scheme, explain why the two parties converge to a common key, and why an attacker using a similar neural network is unlikely to converge to the same key. However, in the second part of the paper we show that this key exchange protocol can be broken in three different ways, and thus it is completely insecure.

## 1 Introduction

Neural networks have attracted a lot of attention in the last 60 years as a plausible computational model of how the human brain operates. The model was first formalized in 1943 by Warren McCulloch and Walter Pitts, and in 1949 Donald Hebb published his highly influential book *Organization of Behavior* in which he studied a variety of neural learning mechanisms. Today the area continues to be extremely active, and attracts interdisciplinary researchers from a wide variety of backgrounds (Biology, Medicine, Psychology, Physics, Mathematics, Computer Science, etc).

Not surprisingly, researchers have also tried to use neural networks in Cryptography. In January 2002, the Physicists Kanter, Kinzel and Kanter [3] proposed a new key exchange protocol between two parties  $\mathcal{A}$  and  $\mathcal{B}$ . It uses the new notion of chaotic synchronization, which makes it possible for two weakly interacting chaotic systems to converge even though each one of them (viewed individually) continues to move in a chaotic way. Many papers and several conferences were devoted in the last few years to this subject, and an excellent starting point for this literature can be found in [5].

In this paper we analyse the Kanter, Kinzel and Kanter (KKK) proposal, which can be viewed as a gradual type of Diffie Hellman key exchange. In both schemes the two parties start from random uncorrelated  $t$ -bit states. The DH scheme uses a single round in which each party reveals  $t$  bits of information about its state. Based on the received information, each party modifies its state once, and the new states become identical. The KKK scheme uses multiple (typically  $\geq t$ ) rounds in which each party reveals a single bit of information about its current state, and then modifies its state according to the information revealed

by the other party. If we denote the sequence of states of the two parties by  $\mathcal{A}_i$  and  $\mathcal{B}_i$ , then  $\text{distance}(\mathcal{A}_{i+1}, \mathcal{B}_{i+1}) < \text{distance}(\mathcal{A}_i, \mathcal{B}_i)$  and eventually  $\mathcal{A}_i = \mathcal{B}_i$  for all  $i \geq i_0$ . However, the parties do not converge by moving towards a common fixedpoint which is halfway between them, and both  $\text{distance}(\mathcal{A}_{i+1}, \mathcal{A}_i)$  and  $\text{distance}(\mathcal{B}_{i+1}, \mathcal{B}_i)$  remain large even for  $i \geq i_0$ . From the point of view of the cryptanalyst the states of the parties become rapidly moving targets, and his main problem is how to combine bits of information about two converging sequences of unknown states. Such multi-round synchronization is a new and unexplored idea, which makes it possible to use new types of cryptographic functions which are not based on number theory. A KKK-like scheme can thus provide a new basis for security, and can give rise to potentially faster key exchange schemes.

The concrete proposal in [3] uses two neural networks in which each network tries to learn from the other network's outputs on common random inputs. In the standard learning problem a neural network tries to learn a fixed function, and thus it converges towards it as a fixedpoint. In the mutual learning problem each network serves both as a trainer and as a trainee, and there is no fixed target to converge to. Instead, they chase each other in a chaotic trajectory which is driven primarily by the common sequence of random inputs. We first consider the nontrivial issue of why the scheme works at all, i.e., why the two chaotic behaviours become synchronized. We then explain the empirical observation in [3] that an attacker who uses an identical neural network with the same learning procedure is extremely unlikely to synchronize his network with the other parties' network even though he eavesdrops to all their communication. However, in the last part of the paper we show that the KKK scheme can be broken by at least three different attacks (using genetic algorithms, geometric considerations, and probabilistic analysis). The bottom line of our analysis is that even though this concrete cryptographic scheme is insecure, the notion of chaotic synchronization is an exciting new concept and a potential source of new insights into how parties can agree on common secret values as a result of public discussion.

## 2 The KKK Key Exchange Scheme

Each party in the proposed KKK construction uses a two level neural network: The first level contains  $K$  independent perceptrons, while the second level computes the parity of their  $K$  hidden outputs. Each one of the  $K$  perceptrons has  $N$  weights  $w_{k,n}$  (where  $1 \leq k \leq K$  and  $1 \leq n \leq N$ ). These weights are integers in the range  $\{-L, \dots, L\}$  that can change over time. Given the  $N$  bit input  $(x_{k,1}, \dots, x_{k,N})$  (where  $x_{k,n} \in \{-1, +1\}$ ), the perceptron outputs the sign (which is also in  $\{-1, +1\}$ ) of  $\vec{w}_k \cdot \vec{x}_k = \sum_{n=1}^N w_{k,n} x_{k,n}$ . The output  $o_k$  of the perceptron has a simple geometric interpretation: the hyperplane which is perpendicular to the weight vector  $\vec{w}$  divides the space into two halves, and the output of the perceptron for input  $\vec{x}$  indicates whether  $\vec{x}$  and  $\vec{w}$  are on the same side of this hyperplane or not (i.e., whether the angle between  $\vec{w}$  and  $\vec{x}$  is less than or greater than 90 degrees). The output of the neural network is defined as the parity  $O = \prod_{k=1}^K o_k$  of the outputs of the  $K$  perceptrons.

In the KKK scheme the two parties  $\mathcal{A}$  and  $\mathcal{B}$  start from random uncorrelated weight matrices  $\{w_{k,n}\}$ . At each round a new random matrix of inputs  $\{x_{k,n}\}$  is publicly chosen (e.g., by using a pseudo random bit generator), and each party announces the output of its own neural network on this common input. If the two output bits are the same, the parties do nothing and proceed to the next round; otherwise each party trains its own neural network according to the output of the other party. The training uses the classical Hebbian learning rule to update the perceptron weights. However, each party knows only the parity (rather than the individual values) of the outputs of the other party's perceptrons, and thus the learning rule has to be modified: In the KKK proposal (which is also discussed and justified in [1] and [4]) each party only modifies those perceptrons in his own network whose hidden outputs differ from the announced output. With this correction, KKK observed that for some choices of  $K$ ,  $N$  and  $L$ , the weight matrices of the two parties become anti parallel (i.e.,  $w_{k,n}^{\mathcal{A}} = -w_{k,n}^{\mathcal{B}}$  for all  $k$  and  $n$ ) after a reasonably small number of rounds, and from then on they always generate negated outputs and always update their weights into new anti parallel states. The two parties could become aware of the achieved synchronization by noticing that their announced outputs were always negated for 20-30 consecutive steps. Once their networks become synchronized, the two parties could stop and compute a common cryptographic key by hashing their current weight matrix (or its negation).

### 3 The Synchronization Process

In this section we explain the synchronization process by using some elementary properties from the theory of random walks in bounded domains. In particular, we analyse the effect of various choices of parameters on the rate of convergence.

The standard Hebbian learning rule forces the mutually learning neural networks into anti parallel states. This is unintuitive, complicates our notation, and makes it difficult to prove convergence by using distance arguments. We thus modify the original scheme in [3], and update the two weight matrices whenever the networks agree (rather than disagree) on some input. We modify several other minor elements, and get a dual scheme in which one of the parties goes through the same sequence of states and the other party goes through a negated sequence of intermediate steps, compared to the original KKK proposal. In this dual scheme the two parties eventually becomes identical (rather than anti parallel). For  $K = 3$ , the modified learning procedure is defined in the following way: Given random public vectors  $\vec{x}_1, \vec{x}_2, \vec{x}_3 \in \{-1, 1\}^N$ , each party calculates its perceptrons' hidden outputs  $o_1 = \text{sgn}(\vec{w}_1 \vec{x}_1)$ ,  $o_2 = \text{sgn}(\vec{w}_2 \vec{x}_2)$ ,  $o_3 = \text{sgn}(\vec{w}_3 \vec{x}_3)$ , where  $\text{sgn}(x)$  is 1 if  $x \geq 0$  and -1 otherwise. It then announces its final output  $O = o_1 o_2 o_3$ . If  $O^{\mathcal{A}} \neq O^{\mathcal{B}}$  the parties end the current round without changing any weights. Otherwise, each party updates only perceptrons for which  $o_k = O$  (since the common  $O$  is the product of three hidden values, each party updates the weights of either one or three perceptrons). The updated weights of perceptron  $k$  are defined by the transformation  $\vec{w}_k \leftarrow \text{bound}_{-L,L}(\vec{w}_k - o_k \vec{x}_k)$ , where

$\text{bound}_{-L,L}$  changes any coordinate which exceeds the allowed weight bounds back to the bound (i.e.,  $-L - 1$  is changed to  $-L$  and  $L + 1$  is changed to  $L$ ).

This learning procedure is quite delicate, and changing the identity of the updated perceptrons or the way they are updated either destroys the synchronization process or makes it trivially insecure. The goal of this section is to explain why the two parties converge, and why a third neural network cannot converge to the same weight matrix by following the same learning procedure.

We first consider a highly simplified neural network which consists of a single perceptron with a single weight. The convergence of such networks is completely explained by the existence of the absorbing boundaries  $-L$  and  $L$  for the range of allowed weight values. Let  $a_i$  and  $b_i$  be the current weights of the two perceptrons. At each round a new random input  $x_i \in \{-1, 1\}$  is chosen, and the parties decide to either ignore it, or to simultaneously move their two weights in the same direction determined by  $x_i$ . However, if any weight tries to step beyond the allowed boundaries, it remains stuck at the boundary while the other weight moves towards it (unless it is also stuck at the same boundary). Each weight starts from a random value, and performs a one dimensional random walk which is driven by the common sequence of random inputs. When neither one of the weights is stuck at the boundary, their distance remains unchanged ( $|a_{i+1} - b_{i+1}| = |a_i - b_i|$ ), whereas if one of them is stuck and the other one moves, their distance is reduced by one. Since each random walk is likely to hit the boundary infinitely often, their distance will eventually reduce to zero, and from then on the two random walks will always coincide.

The case of a single perceptron with multiple weights is a simple generalization of this case. The two weight vectors move in the same direction determined by  $\vec{x}_i$  in a bounded multidimensional box, and along each coordinate the distance is either preserved or reduced by one. When all these distances are reduced to zero, the two random walks become identical forever.

Unfortunately, single perceptron neural networks can be trivially attacked by any neural network which starts from a random initial state and mimics the operation of the two parties. In fact, except during a short initial period, the state of all these perceptrons is uniquely determined by the (publicly known) sequence of inputs  $\vec{x}_i$ , and is independent of their initial state. Consequently, the synchronization process of single perceptron neural networks is trivial, and cannot be used to derive a cryptographically secure common key.

The case of neural networks with multiple perceptrons is more complicated, since the two parties may update different subsets of their perceptrons in each round. We thus have to consider a noisy version of the previous convergence argument, in which occasionally the parties perform *uncoordinated moves* which add  $\vec{x}_i$  to one of  $\mathcal{A}$ 's perceptrons but adds zero to the corresponding perceptron of  $\mathcal{B}$ , which can either increase or decrease the distance between them. Initially there is some weak correlation between  $o_k^{\mathcal{A}}$  and  $o_k^{\mathcal{B}}$  due to the asymmetry in  $o$  caused by cases in which  $\vec{w}_k \vec{x}_k = 0$ . If the parties make a coordinated move (i.e.  $o_k^{\mathcal{A}} = o_k^{\mathcal{B}}$ ) then  $\vec{w}_k^{\mathcal{A}}$  and  $\vec{w}_k^{\mathcal{B}}$  become closer to each other and thus  $\vec{w}_k^{\mathcal{A}} \vec{x}_k$  and  $\vec{w}_k^{\mathcal{B}} \vec{x}_k$  will have an increased tendency to have the same sign (and thus make a

coordinated move) in the next round with a new random input. In particular, if  $\vec{w}_k^{\mathcal{A}} = \vec{w}_k^{\mathcal{B}}$  for all  $k$  then all their future moves will be coordinated, and thus their weight matrices will remain identical forever. The convergence argument becomes a delicate balance between the reduced distance caused by coordinated moves, the increased distance which may be caused by uncoordinated moves, and the probability of making an uncoordinated move as a function of the current correlation between the weights: If we completely rerandomize the weights whenever the two perceptrons make an uncoordinated move the parties will never converge, but if we do not penalize such failures then any third party will also converge to the same state in the same amount of time.

The claimed basis for the security of the scheme in [3] is the proven fact that given fewer than some number  $\alpha(L)N$  of outputs of a parity machine with fixed weights for random inputs it is information theoretically impossible to calculate these weights, and the case of changing weights seems to be even harder. However, the problem of computing the initial weights and the problem of finding the final weights are completely different, and the attacker is only interested in the latter problem. To illustrate this point, consider the simple example of a one dimensional random walk with boundaries. Although it is information theoretically impossible to recover the initial position  $a_0$  from an arbitrarily longer sequence of state signs, it is easy to predict with overwhelming probability all the states from some point onwards.

The other evidence of security given in [3] was the fact that an attacker using the same neural network and a variety of learning rules failed to converge to the same states in the same number of steps as the two parties (in some cases the attacker never converged, and in other cases its convergence was so slow that when the two parties stopped revealing their output bits its state was still completely different). This is a necessary condition for the security of the scheme, but far from being sufficient. However, the cause of this failure is not obvious, and its analysis is very instructive.

Consider an attacker  $\mathcal{C}$  who starts from the same parity machine with randomly chosen weights. At each step she computes her hidden outputs  $o_1^{\mathcal{C}}, \dots, o_K^{\mathcal{C}}$  with respect to the publicly available input. If the parties announce different public outputs  $O^{\mathcal{A}} \neq O^{\mathcal{B}}$ ,  $\mathcal{C}$  knows that  $\mathcal{A}$  and  $\mathcal{B}$  do not update their weights, and thus she also skips the current round without updating her weights. If  $O^{\mathcal{A}} = O^{\mathcal{B}}$  then  $\mathcal{C}$  tries to mimic the behavior of  $\mathcal{A}$  and  $\mathcal{B}$  by guessing which perceptrons should be updated, and she uses her hidden outputs to do so using the same rule as the two parties. In [3] it was empirically observed that this strategy does not allow  $\mathcal{C}$  to converge even if she starts from a state which is strongly correlated to that of  $\mathcal{B}$ . In order to understand why  $\mathcal{C}$  fails while  $\mathcal{A}$  and  $\mathcal{B}$  succeed, we have to compare the probability that  $\mathcal{B}$  and  $\mathcal{C}$  make the same update as  $\mathcal{A}$ . Consider for example a neural network with  $K = 2$ , i.e. each party has two perceptrons. Let's define  $p_k = \Pr[o_k^{\mathcal{A}} = o_k^{\mathcal{B}}]$  for random inputs  $\vec{x}_k$ , and for the sake of simplicity assume that it is the same for both perceptrons ( $p_1 = p_2 = p$ ), and for both pairs  $(\mathcal{A}, \mathcal{B})$  and  $(\mathcal{A}, \mathcal{C})$  (note that the outputs of different units are independent since they are functions of independent random inputs). There are four possible



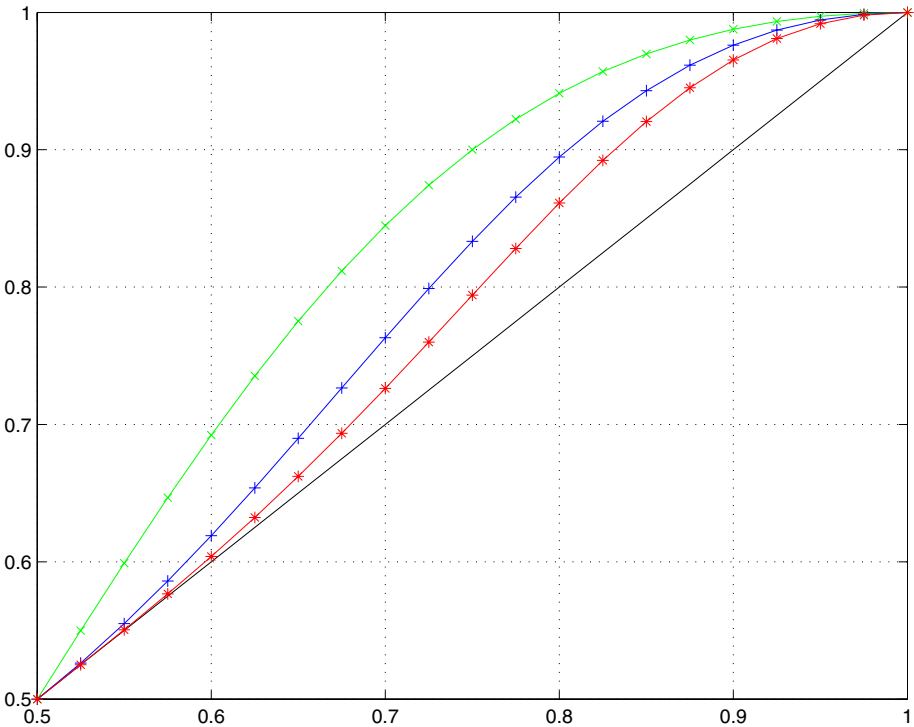
scenarios:  $(o_0^A = o_0^B, o_1^A = o_1^B)$ ,  $(o_0^A \neq o_0^B, o_1^A = o_1^B)$ ,  $(o_0^A = o_0^B, o_1^A \neq o_1^B)$  and  $(o_0^A \neq o_0^B, o_1^A \neq o_1^B)$ , with probabilities  $p^2$ ,  $p(1-p)$ ,  $(1-p)p$  and  $(1-p)^2$  respectively. Note that in the second and the third scenarios  $O^A \neq O^B$ , and thus they will never happen in a round in which  $\mathcal{A}$  and  $\mathcal{B}$  decide to update their weights. However, such scenarios are possible for  $(\mathcal{A}, \mathcal{C})$ , since their outputs can be different when  $\mathcal{C}$  is forced to move, and from her perspective it is a bad idea either to keep the weights unchanged or to update the wrong collection of perceptrons. In other words, the crucial difference between the two parties and the attacker is that the parties can choose the most beneficial points in time at which to update their weights, whereas the passive eavesdropper cannot influence this choice. Consequently, the probability that  $(\mathcal{A}, \mathcal{B})$  make a coordinated move is  $\frac{p^2}{p^2 + (1-p)^2}$ , while the probability that  $(\mathcal{A}, \mathcal{C})$  make a coordinated move is  $p$ . Since  $0 < p < 1$ , it is easy to see that  $\frac{p^2}{p^2 + (1-p)^2} > p$ . Figure 1 shows the probability of making a coordinated move as a function of  $p$  for various numbers of perceptrons  $K$ . It is clear from this figure that the choice of  $K = 2$  is optimal for  $(\mathcal{A}, \mathcal{B})$ . We already demonstrated a difference of behaviour between the  $(\mathcal{A}, \mathcal{B})$  and  $(\mathcal{A}, \mathcal{C})$  cases, but in order to show why such a difference allows the pair  $(\mathcal{A}, \mathcal{B})$  to converge with very high probability but the pair  $(\mathcal{A}, \mathcal{C})$  to converge only with negligible probability, we have to consider the *speed* of convergence.

First we have to define the notion of closeness between perceptrons:  $\rho(\vec{w}, \vec{w}') = \Pr_x[\text{sgn}(\vec{w}\vec{x}) = \text{sgn}(\vec{w}'\vec{x})]$  for a random input  $\vec{x}$  (by definition,  $0 \leq \rho \leq 1$ ). To calculate the expected change of  $\rho$  after one round in which the parties update their weights, we use the following formula:

$$E[\Delta\rho] = \Pr[\top]\Delta\rho_{\top} + \Pr[\perp]\Delta\rho_{\perp},$$

where we use  $\top$  to denote coordinated moves (in which the hidden outputs are the same) and  $\perp$  to denote uncoordinated moves. For the sake of simplicity consider again the case of  $K = 2$  and  $\rho(\vec{w}_1, \vec{w}'_1) = \rho(\vec{w}_2, \vec{w}'_2)$ . Using a large number of numerical experiments we found the forms of  $\rho'_{\top}$  and  $\rho'_{\perp}$ . The results are shown in figure 2, which describes the closeness before and after a coordinated and an uncoordinated move in the various experiments. In order to combine these results we approximated  $\Delta\rho_{\top}$  and  $\Delta\rho_{\perp}$  by two third degree polynomials which are described in figure 3. Using this approximation, figure 4 shows the expected increase of  $\rho$  as a function of the current value of  $\rho$  for the whole system.

Using figure 4 we can easily explain why the pair  $(\mathcal{A}, \mathcal{B})$  quickly converges:  $\Delta\rho(\vec{w}^A, \vec{w}^B) > 0$ , so each step is expected to increase  $\rho$  until eventually  $\rho = 1$ . However, for  $(\mathcal{A}, \mathcal{C})$  the drift is positive only before approximately 0.8, but if  $\mathcal{C}$  gets any closer then her drift is negative and thus her strategy is counterproductive. This explains the experimental result described in [3] — even if  $\rho(\vec{w}^B, \vec{w}^C)$  is relatively high it tends to decrease, and thus such an adversary has a negligible probability to converge to the common states of  $\mathcal{A}$  and  $\mathcal{B}$ .



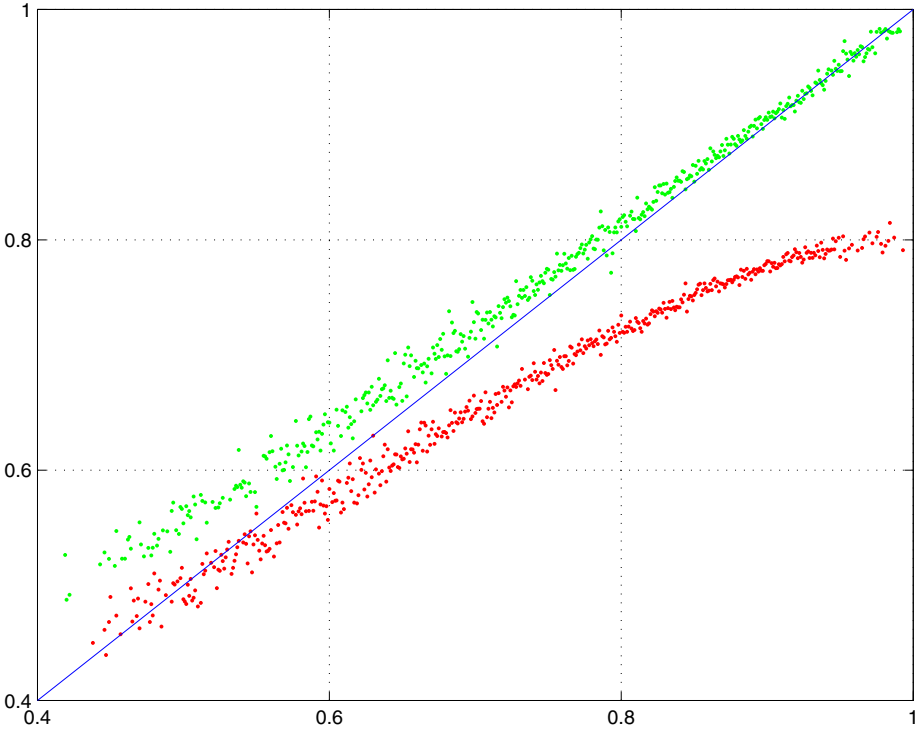
**Fig. 1.** The probability of making a coordinated move for one perceptron when  $K = 2$  (“x”),  $K = 3$  (“+”), or  $K = 4$  (“\*”) (the diagonal line is the identity function  $y = x$ )

## 4 Cryptanalytic Attacks

The security of the scheme was analysed in [3] in terms of its robustness against a particular attacker who simulates the actions of the two parties. The security of the scheme against such an attack was experimentally verified in [3], and mathematically explained in the previous section. In this section we consider other types of attacks, and show that the KKK scheme can be broken by three completely different cryptanalytic techniques. Since the proposed cryptographic scheme is very different from standard schemes, the attacks are also somewhat unusual.

### 4.1 The Genetic Attack

Since the cryptosystem is based on the biological notion of neural networks, we decided to apply a biologically motivated attack based on genetic algorithms. The general idea behind any genetic algorithm is to simulate a population of virtual organisms and to impose evolutionary rules which prefer organisms with certain desirable properties. The literature contains very few successful cryptanalytic



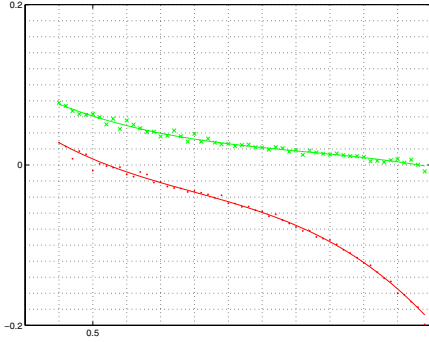
**Fig. 2.** Experimental form of  $\rho'_\perp$  (the lower distribution) and  $\rho'_\top$  (the upper distribution) for  $L = 3$  and  $N = 101$ .

applications of such techniques, but a recent exception is the simulated annealing attack on the PPP scheme described in [2].

In our attack, we simulate a large population of neural networks with the same structure as the two parties, and train them with the same inputs. At each stage about half the simulated networks announce an output of  $+1$ , and half announce an output of  $-1$ . Networks whose outputs mimic those of the two parties breed and multiply, while unsuccessful networks die.

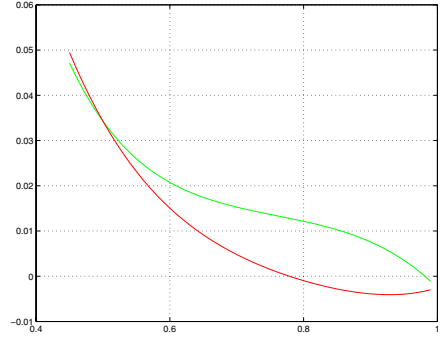
We start the attack with one network with randomly chosen weights. At each step a population of networks evolves according to three possible scenarios:

- $\mathcal{A}$  and  $\mathcal{B}$  have different outputs  $O^{\mathcal{A}} \neq O^{\mathcal{B}}$ , and thus do not change their weights. Then all the attacker's networks remain unchanged as well.
- $\mathcal{A}$  and  $\mathcal{B}$  have the same outputs  $O^{\mathcal{A}} = O^{\mathcal{B}}$ , and the total number of attacking networks is smaller than some limit  $M$ . In this case there are 4 possible combinations of the hidden outputs agreeing with the final output. So, the attacker replaces each network  $\mathcal{C}$  from the population by 4 variants of itself,  $\mathcal{C}_1, \dots, \mathcal{C}_4$ , which are the results of updating  $\mathcal{C}$  with the standard learning



**Fig. 3.** Approximation of  $\Delta\rho_{\perp}$  (the lower distribution) and  $\Delta\rho_{\top}$  (the upper distribution).

The polynomials are  $f_c(p) = -0.6364p^3 + 1.5516p^2 - 1.3409p + 0.4231$  and  $f_n(p) = -1.8666p^3 + 3.6385p^2 - 2.5984p + 0.6304$ .



**Fig. 4.** The speed of convergence for  $(\mathcal{A}, \mathcal{B})$  (the upper line) and  $(\mathcal{A}, \mathcal{C})$  (the lower one) for  $L=3$ ,  $N=101$  and  $K=2$ .

rule but pretending that the hidden outputs were equal to each one of these combinations.

- $\mathcal{A}$  and  $\mathcal{B}$  have the same outputs  $O^{\mathcal{A}} = O^{\mathcal{B}}$  but the total number of simulated networks is larger than  $M$ . In this case the attacker computes the outputs of all the networks, deletes the unsuccessful networks whose output is different from  $O^{\mathcal{A}}$ , and updates the weights in the successful networks by using the standard learning rule with the actual hidden outputs of the perceptrons.

Shortly after  $\mathcal{A}$  and  $\mathcal{B}$  synchronize for the first time, they know this fact, and the attacker uses the same test to check whether any one of his networks has the same weights as  $\mathcal{A}$ . For the recommended choice of parameters ( $K = 3$ ,  $N = 101$ ,  $L = 3$ ), we tried the attack with a threshold of  $M = 2500$  networks, and in more than 50% of our tests at least one of the attacking networks  $\mathcal{C}$  became synchronized with  $\mathcal{A}$  even before  $\mathcal{A}$  and  $\mathcal{B}$  themselves became fully synchronized.

We successfully applied this attack to several variants of the KKK scheme using different parameters as well as different rules for updating the weights and computing the output. The attack is particularly effective for variants in which the genetic attack has a small local branching rate (e.g., when  $K = 2$ ).

## 4.2 The Geometric Attack

We have already described the geometric interpretation of the action of a perceptron. Now we are going to exploit this characterization in order to gain useful information about the unknown weights of neural networks which are defined as the parity of several perceptrons.

Each input can be viewed as  $K$  random hyperplanes  $X_1, \dots, X_K$  corresponding to  $K$  perceptrons. Each  $X_i$  is a hyperplane

$$f_i(z_1, \dots, z_N) = \sum_{j=1}^N x_{ij} \cdot z_j = 0$$

in the  $N$ -dimensional discrete space  $\mathbf{U} = \{-L, \dots, L\}^N$ . The weights of a network could be also viewed as  $K$  points  $W_1, \dots, W_K$  in  $\mathbf{U}$ ,  $W_i = (w_{i1}, \dots, w_{iK})$ , while the  $i$ -th hidden output is just the side of the half-space (with respect to  $X_i$ ) which contains  $W_i$ .

Consider an attacking network  $\mathcal{C}$  that is close enough to the unknown network  $\mathcal{A}$  but has a different output for a given input. In fact they have either 1 or 3 different hidden outputs. The second case is less likely to occur so we assume that only one hidden output of the network  $\mathcal{C}$  is different from the corresponding hidden output of  $\mathcal{A}$ . Consequently, only one pair  $(W_i^{\mathcal{A}}, W_i^{\mathcal{C}})$  is separated by the known input hyperplane  $X_i$ . Of course, we are interested in detecting its index  $i$ .

If the points  $W_i^{\mathcal{C}}$  and  $W_i^{\mathcal{A}}$  are separated by  $X_i$  then the distance between them is greater than the distance from  $W_i^{\mathcal{C}}$  to the hyperplane  $X_i$ .  $W_i^{\mathcal{C}}$  and  $W_i^{\mathcal{A}}$  are close to each other, so the distance from  $W_i^{\mathcal{C}}$  to  $X_i$  has to be small. On the other hand, if  $W_i^{\mathcal{C}}$  and  $W_i^{\mathcal{A}}$  are in the same half-space with respect to  $X_i$  then they are more likely to be far away from the random input  $X_i$  (even though we know that they are close to each other). We thus guess that the index of the incorrect hidden output is the  $i$  for which  $W_i^{\mathcal{C}}$  is closest to the corresponding hyperplane  $X_i$ , where we compute the distance by  $\rho(W_i^{\mathcal{C}}, X_i) = |f_i(W_i^{\mathcal{C}})|$ .

Formally, the attacker constructs a single neural network  $\mathcal{C}$  with the same structure as  $\mathcal{A}$  and  $\mathcal{B}$ , and randomly initializes its weights. At each step she trains  $\mathcal{C}$  with the same input as the two parties, and updates its weights with the following rules:

- If  $\mathcal{A}$  and  $\mathcal{B}$  have different outputs  $O^{\mathcal{A}} \neq O^{\mathcal{B}}$ , then the attacker doesn't update  $\mathcal{C}$ .
- If  $\mathcal{A}$  and  $\mathcal{B}$  have the same outputs  $O^{\mathcal{A}} = O^{\mathcal{B}}$  and  $O^{\mathcal{C}} = O^{\mathcal{A}}$ , then the attacker updates  $\mathcal{C}$  by the usual learning rule.
- If  $\mathcal{A}$  and  $\mathcal{B}$  have the same outputs  $O^{\mathcal{A}} = O^{\mathcal{B}}$  and  $O^{\mathcal{C}} \neq O^{\mathcal{A}}$ , then the attacker finds  $i_0 \in \{1, \dots, K\}$  that minimizes  $|\sum_{j=0}^N w_{ij}^{\mathcal{C}} \cdot x_{ij}|$ . The attacker negates  $o_{i_0}^{\mathcal{C}}$  and updates  $\mathcal{C}$  assuming the new hidden bits and output  $O^{\mathcal{A}}$ .

Different attackers starting from randomly chosen states behave independently and thus multiple attackers have a higher probability to be successful. We tried this attack with 100 random initial states and at least one of them synchronized with  $\mathcal{A}$  faster than  $\mathcal{B}$  with probability  $> 90\%$ .

### 4.3 The Probabilistic Attack

As was described in the previous section, it is much easier to predict the position of a point in a bounded multidimensional box after several moves in its random

walk than to guess its original position. A simple way to do it is to consider each coordinate separately, and to associate with each possible value  $i$  in the interval  $\{-L, \dots, L\}$  the probability  $p_t(i) = \Pr[x_t = i]$ . Initially  $\forall i, p_0(i) = \frac{1}{2L+1}$  and after each move  $p_{t+1}(i) = \sum_j p_t(j)$ , where  $j$  are such that if  $x_t = j$  then  $x_{t+1} = i$ . Applying this technique to the original scheme we face the problem that the moves are not known — the attacker does not know which perceptrons are updated in each round. Fortunately, if we know the distribution of the probabilities  $P_{k,n,i} = \Pr[w_{k,n} = i]$  then using dynamic programming we can calculate the distribution<sup>1</sup> of  $\vec{w}_k \vec{x}_k$  for a given vector  $\vec{x}_k$  and thus the probabilities  $u_k(s) = \Pr[o_k = s]$ . Using these probabilities we can calculate the conditional probabilities  $U_k = \Pr[o_k = 1|O]$ :

$$U_k = \frac{\sum_{(\alpha_1, \dots, \alpha_K): \prod_i \alpha_i = O \& \alpha_k = 1} \prod_i u_i(\alpha_i)}{\sum_{(\alpha_1, \dots, \alpha_K): \prod_i \alpha_i = O} \prod_i u_i(\alpha_i)},$$

because  $O$  is publicly known. We can now update the distribution of the weights:  $P_{k,n,i}^{t+1} = \sum_j P_{k,n,j}^t \Pr[w_{k,n}^t = j \Rightarrow w_{k,n}^{t+1} = i]$ , where  $\Pr[w_{k,n}^t = j \Rightarrow w_{k,n}^{t+1} = i]$  is calculated using  $U_k$ . Experiments show that in most cases, when  $\mathcal{A}$  and  $\mathcal{B}$  converge to a common  $\hat{w}_{k,n}$ , the probabilities  $\Pr[w_{k,n} = \hat{w}_{k,n}] \approx 1$  and thus the adversary can easily find  $\hat{w}_{k,n}$  when  $\mathcal{A}$  and  $\mathcal{B}$  decide to stop the protocol.

## References

1. M. Biehl, N. Caticha, “Statistical Mechanics of On-Line Learning and Generalization”, The Handbook of Brain Theory and Neural Networks, 2001.
2. John A. Clark, Jeremy L. Jacob, “Fault Injection and a Timing Channel on an Analysis Technique”, Proceedings of Eurocrypt 2002, p. 181.
3. Ido Kanter, Wolfgang Kinzel, Eran Kanter, “Secure exchange of information by synchronization of neural networks”, Europhys., Lett. 57, 141, 2002.
4. M. Oppen, W. Kinzel, “Statistical Mechanics of Generalization”, Models of Neural Networks III, 151-20, 1995.
5. <http://rfic.ucsd.edu/chaos>

<sup>1</sup> Note that when we calculate the distribution of  $\vec{w}_k \vec{x}_k$ , we assume that the random variables  $w_{k,n}$  are independent. This seems to be true for the original scheme.

# The Hardness of Hensel Lifting: The Case of RSA and Discrete Logarithm

Dario Catalano, Phong Q. Nguyen, and Jacques Stern

École normale supérieure, Département d'informatique,  
45 rue d'Ulm, 75230 Paris Cedex 05, France,  
{Dario.Catalano,Phong.Nguyen,Jacques.Stern}@ens.fr

**Abstract.** At ACM CCS '01, Catalano *et al.* proposed a mix of the RSA cryptosystem with the Paillier cryptosystem from Eurocrypt '99. The resulting scheme, which we call RSAP, is a probabilistic cryptosystem which is both semantically secure under an appropriate decisional assumption and as efficient as RSA, but without the homomorphic property of the Paillier scheme. Interestingly, Sakurai and Takagi presented at PKC '02 a proof that the one-wayness of RSAP was equivalent to the RSA assumption. However, we notice in this paper that the above proof is not completely correct (it works only in the case when a perfect oracle - i.e. an oracle that always provides correct answers - is given). We fix the proof by presenting a new proof based on low-dimensional lattices. The new proof, inspired by the work of Sakurai and Takagi, is somewhat related to Hensel lifting and the  $N$ -adic decomposition of integer exponentiation. Roughly speaking, we consider the problem of computing  $f(x) \bmod M^\ell$  given  $f(x) \bmod M$  and an exponent  $\ell > 1$ . By studying the case  $f(x) = x^e$  and  $M$  is an RSA-modulus, we deduce that the one-wayness of RSAP is indeed equivalent to the RSA assumption, and we are led to conjecture that the one-wayness of the original Paillier scheme may not be equivalent to the RSA assumption with exponent  $N$ . By analogy, we also study the discrete logarithm case, namely when  $f(x) = g^x$  and  $M$  is a prime, and we show that the corresponding problem is curiously equivalent to the discrete logarithm problem in the subgroup spanned by  $g$ .

**Keywords:** Public-key, RSA, Paillier, Discrete logarithm, Hensel, One-wayness, Lattice.

## 1 Introduction

Many basic computational problems in number theory can be efficiently solved by first looking at the problem modulo a (small) prime number  $p$  and then performing a so-called Hensel lifting, which iteratively transforms solutions modulo  $p$  into solutions modulo arbitrary powers of  $p$ . This is for instance the case with factorization of univariate integer polynomials, and with integer root finding of univariate integer polynomials (see [1, 5]). The lifting process has been dubbed Hensel lifting because of the pioneering work of the German mathematician

Hensel on  $p$ -adic numbers at the end of the 19th century. The  $p$ -adic numbers are beyond the scope of this paper, and we refer the interested reader to [8] for more information: Let us just briefly mention that, mathematically speaking, the  $p$ -adic numbers are an extension (depending on  $p$ ) of the field  $\mathbb{Q}$  of rational numbers, which is built as a completion of  $\mathbb{Q}$  with respect to a specific metric (different from the usual absolute valuation  $|x - y|$ ) related to the decomposition in base  $p$  of every positive integer. The link between Hensel lifting and  $p$ -adic numbers is natural: Hensel lifting produces solutions modulo increasing powers of  $p$  which can be viewed as better and better approximations of some “true” solution, where the quality of the approximation is measured thanks to the specific metric of the  $p$ -adic numbers.

In this paper we consider Hensel lifting from a cryptographic perspective. We study the hardness of the general problem of computing  $f(x) \bmod M^\ell$  (where  $\ell$  is an integer  $\geq 2$ ) given  $f(x) \bmod M$ , where the function  $f$  is implemented as either the RSA function or the Discrete Logarithm function. More precisely, we investigate the following problems:

1. Given an RSA modulus  $N$  and the value  $x^e \bmod N$  where  $0 \leq x < N$ , how hard is it to compute  $x^e \bmod N^\ell$  (for  $\ell > 1$ )?
2. Given a prime  $p$ , an integer  $g$  and the value  $g^x \bmod p$  where  $x$  is defined modulo the order of  $g$ , how hard is it to compute  $g^x \bmod p^\ell$  (again for  $\ell > 1$ )?

**MOTIVATION AND PREVIOUS WORK.** At Eurocrypt '99 Paillier [11] proposed a new cryptosystem based on a novel computational problem: the composite residuosity class problem. The details of the scheme are given below, for now let us highlight the main contributions of Paillier's construction. Given an RSA modulus  $N$ , the multiplicative group  $\mathbb{Z}_{N^2}^*$  can be partitioned into  $N$  equivalence classes according to the following equivalence relation:  $a, b \in \mathbb{Z}_{N^2}^*$  are equivalent if and only if the product  $ab^{-1}$  is an  $N$ -th residue modulo  $N^2$ , where by  $N$ -residue we intend an element  $x \in \mathbb{Z}_{N^2}^*$  such that there exists  $y \in \mathbb{Z}_{N^2}^*$  satisfying the equation  $x \equiv y^N \bmod N^2$ .

The composite residuosity class problem is then the problem to determine, on input a random value  $w \in \mathbb{Z}_{N^2}^*$  to which class such an element belongs. The one-wayness of Paillier's scheme is provably equivalent to the class problem which turns out to be related but not known to be equivalent to the problem of inverting RSA, when the public encryption exponent is set to  $N$ . The semantic security of Paillier's scheme is provably equivalent to a decisional variant of the class problem. Paillier's paper has sparked a huge amount of research due to its beautiful and original mathematical structure. Moreover the scheme is very attractive for many practical applications because of its *homomorphic* property: given the ciphertexts  $c_1 = \mathbf{ENC}(m_1)$  and  $c_2 = \mathbf{ENC}(m_2)$ , an encryption of  $m_1 + m_2$  can easily be obtained by simply multiplying  $c_1$  and  $c_2$ .

The main drawback of Paillier's scheme is its cost: encryption and decryption cost respectively two and one modular exponentiations, but all the operations are performed modulo  $N^2$ . Moreover the exponents used have all order  $\Omega(N)$ .



To improve the efficiency of the scheme, Catalano *et al.* [4] proposed a mix of Paillier's scheme with the RSA scheme, whose running time is comparable to that of plain RSA, and which is still semantically secure under an appropriate decisional assumption. The new scheme follows from an alternative decryption process for a particular instance of Paillier's scheme, which allows to drastically reduce the size of the encryption exponent. Interestingly enough, even though the modification proposed in [4] only slightly changes the encryption scheme, it deeply influences its mathematical structure. In the following we will refer to the Catalano *et al.* cryptosystem as the *RSA–Paillier* Cryptosystem (RSAP for brevity).

Later, Sakurai and Takagi [12] further studied the properties of the RSA–Paillier scheme and presented a proof that its one-wayness is equivalent to the problem of inverting RSA. Unfortunately, even though the proposed ideas are very appealing, they turn out not to be completely sound from a technical point of view. Specifically they prove that the one-wayness of RSA–Paillier cryptosystem is equivalent to the problem of computing, given a value of the form  $r^e \bmod N$ , the “lifted” value  $r^e \bmod N^2$ . Then the proof proceeds by a standard *reductio ad absurdum* argument: they prove that if one has an oracle to efficiently solve the above lifting problem this oracle could be used to construct an efficient algorithm that computes the least significant bit of RSA (which, in turn, is known to be a hard core predicate [2] for the RSA function [7]). However, as we will show in section 3, the argument is flawed, in the sense that the proposed technique works only for the particular case in which the oracle gives a correct answer with probability 1 (and we will note that another result of [12] related to another variant of RSA suffers from the same flaw). Thus the problem of proving the equivalence between the one-wayness of RSA and the one-wayness of RSA–Paillier remains open for the general case in which the provided oracle answers correctly only for a non-negligible fraction of the inputs.

A variant of the Hensel lifting problem was discussed by Takagi [13], who proposed some efficient variants of RSA using  $N$ -adic expansion.

**OUR RESULTS.** Our contributions can be summarized as follows. First of all we prove that the one-wayness of the RSA–Paillier function is actually equivalent to that of the RSA function. We then turn our attention to the original Paillier's trapdoor function and we prove the following, somehow surprising, results:

1. Given a random RSA modulus  $N$ , computing  $r^N \bmod N^2$  from a value  $r^N \bmod N$  where  $0 \leq r < N$  is as hard as solving the composite residuosity class problem.
2. Given a random RSA modulus  $N$ , computing  $r^N \bmod N^3$  from a value  $r^N \bmod N$  where  $0 \leq r < N$  is as hard as inverting RSA when the public exponent is set to  $N$ .

In some sense, the above results seem to provide an intuitive separation between the Class assumption, introduced by Paillier, and the RSA assumption. This leads us to conjecture that the one-wayness of the Paillier scheme is not equivalent to the RSA assumption with exponent  $N$ .

Our techniques can be generalized to the discrete logarithm function (modulo a prime  $p$ ) as well, and we prove that, under certain conditions, the problem of computing  $g^x \bmod p^\ell$  when  $g, p, h = g^x \bmod p$  are given is equivalent to the problem of computing  $x$ . More precisely, the order  $\omega$  of  $g$  modulo  $p$  is assumed to be prime and publicly known, and the integer  $\ell$  is defined as the unique positive integer such that  $g^\omega \not\equiv 1 \pmod{p^\ell}$  and  $g^\omega \equiv 1 \pmod{p^{\ell-1}}$ .

ROAD MAP. The paper is organized as follows. In Section 2 we provide definitions and notations that are useful for the rest of the paper. Then we quickly describe Paillier's cryptosystem and its variant from [4]. Section 3 presents our results for the RSA case. The discrete logarithm case is discussed in Section 4. We conclude the paper with some remarks and directions for future research in Section 5.

## 2 Preliminaries

NOTATION (Basically quoted from [4]). In the following we denote by  $\mathbb{N}$  the set of natural numbers, by  $\mathbb{R}^+$  the set of positive real numbers, by  $\mathbb{Z}_N$  the ring of integers mod  $N$ , which we identify to the set  $\{0, 1, \dots, N-1\}$ , and by  $\mathbb{Z}_N^*$  its subset of invertible elements. In particular, we view elements of  $\mathbb{Z}_N$  as integers of  $\{0, \dots, N-1\}$ : for instance, if  $r \in \mathbb{Z}_N$ ,  $r^e \bmod N^2$  denotes the integer  $r$  raised to the power  $e$  (as an integer and not as an integer mod  $N$ ), eventually taken modulo  $N^2$ . We say that a function  $\text{negl} : \mathbb{N} \rightarrow \mathbb{R}^+$  is *negligible* iff for every polynomial  $P(n)$  there exists a  $n_0 \in \mathbb{N}$  s.t. for all  $n > n_0$ ,  $\text{negl}(n) \leq 1/P(n)$ . We denote by  $\mathcal{PRIMES}(k)$  the set of primes of length  $k$ . For  $a, b \in \mathbb{N}$  we write  $a \propto b$  if  $a$  is a non zero multiple of  $b$ .

If  $A$  is a set, then  $a \leftarrow A$  indicates the process of selecting  $a$  at random and uniformly over  $A$  (which in particular assumes that  $A$  can be sampled efficiently).

If  $N$  is an RSA modulus (i.e.  $N = pq$  with  $p, q$  primes), then we denote by  $\text{RSA}[N, e]$  the RSA function with exponent  $e$ . In the following we will assume that  $\text{RSA}[N, e]$  is a one-way function, i.e. that given  $N$  of unknown factorization, a public exponent  $e$  and  $\text{RSA}[N, e](x) = x^e \bmod N$ , for random  $x$  it is infeasible to efficiently compute  $x$ . We will refer to this conjecture as the *RSA*[ $N, e$ ] *assumption*.

PAILLIER'S SCHEME. Let  $N = pq$  be an RSA modulus and consider the multiplicative group  $\mathbb{Z}_{N^2}^*$ . Let  $g$  be an element whose order is a multiple of  $N$  in  $\mathbb{Z}_{N^2}^*$ . Paillier [11] defines the following function

$$\mathcal{F}_g : \mathbb{Z}_N^* \times \mathbb{Z}_N \rightarrow \mathbb{Z}_{N^2}^*$$

$$\mathcal{F}_g(r, m) = r^N g^m \bmod N^2$$

and proves the following statements:

- The function  $\mathcal{F}_g$  is a trapdoor permutation. The trapdoor information is the factorization of  $N$ .
- Inverting  $\mathcal{F}_g$  is equivalent to inverting  $\text{RSA}[N, N]$ .

By the first property above, once  $g$  is fixed, for a given  $w \in \mathbb{Z}_{N^2}^*$ , there exists a unique pair  $(m, r)$  such that  $w = r^N g^m \bmod N^2$ . We say that  $m$  is the *class* of  $w$  relative to  $g$ , and we indicate this value with  $Class_g(w)$ . We define the *Computational Composite Residuosity Class Problem* as the problem of computing  $m$  when  $N, g$  and  $w$  are provided. We will assume this to be an intractable problem. More formally, we use the following definition from [3]:

**Definition 1.** We say that computing the function  $Class_g(\cdot)$  is hard if, for every probabilistic polynomial time algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\cdot)$  such that

$$\Pr \left[ \begin{array}{l} p, q \leftarrow \text{PRIMES}(n/2); \ N = pq; \\ g \leftarrow \mathbb{Z}_{N^2}^* \text{ s.t. } \text{ord}(g) \propto N; \\ c \leftarrow \mathbb{Z}_N; \ z \leftarrow \mathbb{Z}_N^*; \ w = g^c z^N \bmod N^2; \\ \mathcal{A}(N, g, w) = c \end{array} \right] = \text{negl}(n)$$

In his paper Paillier proves that the function  $Class$  is random self reducible [2] over  $g \in \mathbb{Z}_{N^2}^*$ , i.e. that its complexity is independent of the specific base  $g$  used.

**THE RSA-PAILLIER SCHEME.** Let  $N = pq$  be an RSA modulus and consider the multiplicative group  $\mathbb{Z}_{N^2}^*$ . For a random  $e \in \mathbb{Z}_N$  such that  $\gcd(e, \lambda(N^2)) = 1$ , Catalano *et al.* [4] defined the following function

$$\mathcal{E}_e : \mathbb{Z}_N^* \times \mathbb{Z}_N \rightarrow \mathbb{Z}_{N^2}^*$$

$$\mathcal{E}_e(r, m) = r^e (1 + mN) \bmod N^2$$

and they proved it is a trapdoor permutation equivalent to  $RSA[N, e]$ .

To encrypt a message  $m$ , one simply chooses a random  $r \in \mathbb{Z}_N^*$  and sets  $c = (1 + mN)r^e \bmod N^2$ . From the ciphertext  $c$ , anyone knowing the factorization of  $N$  can retrieve the message, by first computing  $r = \sqrt[e]{c} \bmod N$  and then getting  $m$  as  $\frac{(c(r^{-1})^e \bmod N^2) - 1}{N}$  over the integers.

Notice that, in order for the above decryption procedure to work it is not necessary to assume  $\gcd(e, \lambda(N^2)) = 1$ . As a matter of fact, one can consider exponents  $e$  such that  $\gcd(e, \lambda(N)) = 1$ .

In this sense by letting  $e = N$  we go back to an instance of Paillier's scheme where  $g$  is set to  $(1 + N)$ . For the purposes of this paper, however, we will assume  $\gcd(e, \lambda(N^2)) = 1$ . The reason for this choice will become clearer in the next section.

### 3 The RSA Case

We start this section by introducing a new computational problem, which is actually very similar to a problem presented in [12].

Informally, the problem we have in mind can be stated as follows. Assume an RSA modulus  $N$  is provided, given  $c = r^e \bmod N$  (where  $r \leftarrow \{0, \dots, N-1\}$ ),

we want to compute the “lifted” value  $r^e \bmod N^\ell$  for  $\ell > 1$ . More formally we define the following function over  $\{0, \dots, N-1\}$ :

$$\mathbf{Hensel - RSA}[N, e, \ell](r^e \bmod N) = r^e \bmod N^\ell$$

Note that this function is well-defined over  $\{0, \dots, N-1\}$  because the RSA-function is a permutation over  $\mathbb{Z}_N$ .

It is immediate to see that if the factorization of the modulus is known then one can efficiently compute **Hensel-RSA**. On the other hand, if the factorization of  $N$  is not available, we conjecture it is infeasible to compute such a function in probabilistic polynomial time.

**Definition 2.** We say that computing the function **Hensel-RSA** $[N, e, \ell](r^e \bmod N)$  is hard if, for every probabilistic polynomial time algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}()$  such that

$$\Pr \left[ \begin{array}{l} p, q \leftarrow \text{PRIMES}(n/2); \ N = pq; \\ r \leftarrow \{0, \dots, N-1\}; \ w = r^e \bmod N; \\ \mathcal{A}(N, e, w, \ell) = r^e \bmod N^\ell \end{array} \right] = \text{negl}(n)$$

In the next lemma (originally presented, in a slightly different form, in [12]) we make explicit the relation existing between the problem of computing the function Hensel-RSA and the one-wayness of the RSA-Paillier scheme.

The proof is straightforward and is left to the reader.

**Lemma 1.** *Given an RSA modulus  $N$  and a public exponent  $e$ , the RSA-Paillier function is one-way if and only if Hensel-RSA $[N, e, 2]$  is hard.*

Now, on top of Lemma 1, we prove that the one-wayness of RSA-Paillier is equivalent to the one-wayness of RSA, by showing that the problem of computing Hensel-RSA, with parameters  $N, e$  and 2, on input  $r^e \bmod N$  and the one of computing  $r$  from  $r^e \bmod N$  are computationally equivalent. Observe that assuming that Hensel-RSA $[N, e, 2]$  is hard implicitly implies that the RSA $[N, e]$  assumption must hold. Consequently, we will focus on proving that the inverse direction also holds, i.e. that under the RSA $[N, e]$  assumption Hensel-RSA $[N, e, 2]$  is hard.

### 3.1 A Flawed Solution

In this paragraph we discuss the approach followed by Sakurai and Takagi [12, Theorem 2], and we show why it is incorrect.

As already sketched in the introduction, they propose the following strategy: assume, for the sake of contradiction, that one has an oracle  $\mathcal{O}$  that, on input  $r^e \bmod N$ , computes  $r^e \bmod N^2$  with some non negligible probability of success  $\varepsilon$ . Then, on input a random RSA ciphertext  $r^e \bmod N$ , the basic idea of their proof is to use such an oracle to compute the least significant bit of  $r$  with some non-negligible advantage, and then apply the bit-security result of [7]. They implement this idea as follows:

1. Run  $\mathcal{O}(r^e \bmod N)$  and obtain  $b_0 + b_1N = r^e \bmod N^2$ .
2. Run  $\mathcal{O}((2^{-1}r)^e \bmod N)$  and obtain  $a_0 + a_1N = (2^{-1}r)^e \bmod N^2$ .
3. Return 1 as the *lsb* of  $r$  if  $a_0 + a_1N = 2^{-e}(b_0 + b_1N) \bmod N^2$  holds and 0 otherwise.

Finally they claim that the success probability of the above algorithm is  $\varepsilon^2$ . However this is not true. As a matter of fact in order for such an estimate to be correct it is crucial to query the oracle on *random* and *independently generated* inputs. Here, on the contrary, the two inputs are clearly not independently sampled. Thus it is not possible to bound by  $\varepsilon^2$  the probability of success of the algorithm<sup>1</sup>. By the way, exactly the same mistake appears in another part of the paper [12], more precisely in the proof of [12, Theorem 6], related to the one-wayness of another class of probabilistic variants of RSA.

Furthermore, we note that even if the proof was correct, the reduction would be rather inefficient in terms of oracle calls. Indeed, the reduction makes two oracle calls to obtain only one bit of information on  $r$ , which implies that to completely recover  $r$ , one has to make at least  $2 \log N$  oracle calls. And one also has to use the reduction of the bit-security result of [7].

### 3.2 Our Solution

With the next theorem we propose a general result connecting the difficulty of computing the Hensel-RSA function with the hardness of inverting RSA. Specifically we prove that, given a public exponent of the form  $e = fN^\ell$  (for constants  $\ell \geq 0$  and  $f > 0$  such that  $\gcd(f, \lambda(N^2)) = 1$ ), Hensel-RSA $[N, e, \ell + 2]$  is hard if and only if RSA $[N, e]$  is hard. Note that any valid RSA public encryption exponent  $e$  can be written in the form  $e = fN^\ell$  (where  $\gcd(f, \lambda(N^2)) = 1$ ), unless  $\gcd(e, N)$  is a non-trivial factor of  $N$ , in which case the public exponent  $e$  would disclose the RSA private key. As already mentioned our proof will focus on showing that under the RSA $[N, e]$  assumption, Hensel-RSA $[N, e, \ell + 2]$  is hard. Interestingly, our reduction only calls the oracle twice, as opposed to at least  $2 \log N$  for the (flawed) one proposed by [12].

**Theorem 1.** *Given an integer  $N$  and an integer  $e$  of the form  $e = fN^\ell$  where  $f$  is coprime with  $\lambda(N^2)$  and  $\ell \geq 0$ , then Hensel-RSA $[N, e, \ell + 2]$  is hard if and only if the RSA $[N, e]$  assumption holds.*

*Proof.* Assume, for the sake of contradiction, that Hensel-RSA $[N, e, \ell + 2]$  is not hard. This means that there exists an oracle  $\mathcal{O}$  that, on input a random challenge  $w = r^e \bmod N$ , computes  $r^e \bmod N^{\ell+2}$  with some non-negligible probability  $\varepsilon$ . Here we will show how to use this oracle to construct a probabilistic polynomial time algorithm  $\mathcal{I}$  that successfully inverts RSA with a polynomially related probability.

<sup>1</sup> For example it may very well happen that the non-negligible set of inputs for which the oracle answers correctly does not contain any couple of the form  $(r^e \bmod N, (2^{-1}r)^e \bmod N)$ , and, in such a case, the success probability of the algorithm would be 0.

Assume that we are given as input a random element  $w = r^e \bmod N$ : our goal is to compute  $r$ . We start by choosing a random  $a$  uniformly in  $\mathbb{Z}_N^*$ . We then call the oracle  $\mathcal{O}$  twice, on inputs  $w$  and  $(a^e w) \bmod N$ . Since the queries  $w$  and  $(a^e w) \bmod N$  are independent and uniformly distributed over  $\mathbb{Z}_N$  (by definition of  $r$ ,  $w$  and  $a$ ), we obtain with probability  $\varepsilon^2$  the integers  $r^e \bmod N^{\ell+2}$  and  $\mu^e \bmod N^{\ell+2}$  where  $\mu$  is defined by  $\mu = ar \bmod N$ .

We may assume that  $r \in \mathbb{Z}_N^*$ , otherwise either  $r = 0$  or we are able to factor  $N$ . Then  $\mu$  is invertible modulo  $N^{\ell+2}$ , and there therefore exists  $z \in \mathbb{Z}_{N^{\ell+1}}$  such that:

$$ar \equiv \mu(1 + zN) \pmod{N^{\ell+2}} \quad (1)$$

Raising to the power  $e = fN^\ell$ , we obtain:

$$a^e r^e \equiv \mu^e (1 + zfN^{\ell+1}) \pmod{N^{\ell+2}}.$$

In this congruence, we know  $a$ ,  $r^e \bmod N^{\ell+2}$  and  $\mu^e \bmod N^{\ell+2}$ : we can thus compute  $zf$  modulo  $N$ . Since  $f$  is coprime with  $N$ , we derive  $z_0 = z \bmod N$ . Taking equation (1) modulo  $N^2$ , we obtain:

$$ar \equiv \mu(1 + z_0N) \pmod{N^2}, \quad (2)$$

where only  $r$  and  $\mu$  are unknowns both in  $\{1, \dots, N-1\}$ .

To complete the proof, we solve this linear congruence by a lattice reduction argument (see for instance the survey [10] for references on lattice theory). Consider indeed the following set

$$L = \{(R, U) \in \mathbb{Z}^2 : aR \equiv U(1 + z_0N) \pmod{N^2}\}.$$

Since  $L$  is a subgroup of  $\mathbb{Z}^2$ ,  $L$  is a lattice, whose dimension is obviously equal to two. The vector  $(r, \mu)$  belongs to  $L$  and to  $[1, N-1]^2$ . Therefore  $L \cap [1, N-1]^2$  is not empty. A classical lattice reduction result (which can be viewed as a particular case of integer programming in fixed dimension, see [9]) then states that one can compute a vector  $(r', \mu') \in L \cap [1, N-1]^2$  in time polynomial in  $\log N$  (because one obviously knows a basis of  $L$  whose size is polynomial in  $\log N$ ). Because  $(r, \mu)$  and  $(r', \mu')$  both belong to  $L$ , equation (2) implies:

$$r\mu' \equiv r'\mu \pmod{N^2}.$$

Since  $r, \mu, r', \mu'$  all lie in  $[1, N-1]$ , the congruence is in fact an equality over  $\mathbb{Z}$ :  $r\mu' = r'\mu$ . From  $r'$  and  $\mu'$ , we can therefore compute the integers  $r$  and  $\mu$  up to a multiplicative factor, namely  $\gcd(r, \mu)$ .

We now show that with overwhelming probability, this gcd will be sufficiently small that it can be exhaustively searched in polynomial time. To see this, notice that the number of pairs  $(\alpha, \beta) \in [0, N-1]^2$  which have a common divisor  $d$  is  $O(N^2/d^2)$  as  $N$  grows, therefore, for any  $B$ , the number of pairs  $(\alpha, \beta) \in [0, N-1]^2$  which have a gcd  $> B$  is at most  $O(\sum_{d>B} N^2/d^2) = O(N^2/B)$ . Since  $\mu$  and  $r$  are both uniformly distributed over  $\mathbb{Z}_N$ , the probability that  $\gcd(\mu, r) \geq (\log N)/\varepsilon$  is  $O(\varepsilon^2/\log N)$  by taking  $B = (\log N)/\varepsilon^2$ . Finally, we

proved that with probability at least  $\varepsilon^2 - O(\varepsilon^2/\log N) = \varepsilon^2(1 - o(1))$  over the choice of  $(a, r)$ , we can compute in polynomial time  $r$  an  $\mu$  up to the factor  $\gcd(r, \mu)$  which is  $\leq (\log N)/\varepsilon^2$ . Thus, we can compute  $r$  in time polynomial in  $\log N$  and  $1/\varepsilon$ , thanks to an exhaustive search over  $\gcd(r, \mu)$ , since the value of  $r$  can be checked with  $w = r^e \bmod N$ .  $\square$

As an immediate consequence of Theorem 1 and Lemma 1, we obtain:

**Corollary 1.** *Given an RSA modulus  $N$  together with a public exponent  $e$  such that  $\gcd(e, \lambda(N^2)) = 1$ , the RSAP encryption function is one-way if and only if  $\text{RSA}[N, e]$  is a one-way function.*

Observe that, by setting  $f = \ell = 1$  in the parameters of Theorem 1, we get that the hardness of Hensel-RSA $[N, N, 3]$  is actually equivalent to that of RSA $[N, N]$ . To complete the picture, with the next theorem we make explicit the relation existing between the one-wayness of Paillier's encryption function and the problem of computing Hensel-RSA with parameters  $N, N, 2$ .

**Theorem 2.** *Given an RSA modulus  $N$ , then Hensel-RSA $[N, N, 2]$  is hard if and only if  $\text{Class}_g$  is hard.*

*Proof.* Since, for all  $g$  such that  $\text{ord}(g) \propto N$ , all the instances of  $\text{Class}_g(\cdot)$  are computationally equivalent, we will prove the theorem for the case in which  $g = 1 + N$  (note that  $1 + N$  has order  $N$  in  $\mathbb{Z}_{N^2}^*$ ).

First assume that a random ciphertext  $c = (1 + mN)r^N \bmod N^2$  is given. Our goal is to compute  $m$  using an oracle that, when receiving an input of the form  $y^N \bmod N$  returns as output the value  $y^N \bmod N^2$ , with probability  $\varepsilon$  (non negligible). Thus when the oracle is given the value  $c \bmod N$ , it will answer  $(\sqrt[N]{c \bmod N})^N \bmod N^2$  with probability  $\varepsilon$ . Note that this value corresponds to  $r^N \bmod N^2$  (Observe that this is true even in the case in which  $r$  is greater than  $N$ ). From  $r^N \bmod N^2$  and  $r^N \bmod N$  it is easy to compute  $m$ .

Conversely, assume we are given an oracle than on input a random  $c \in \mathbb{Z}_{N^2}^*$  computes the class of  $c$  with respect to the base  $(1 + N)$  (again we denote by  $\varepsilon$  the probability of success of the oracle). Now we would like to compute, for a random challenge  $r^N \bmod N$ , the corresponding  $r^N \bmod N^2$ , using the provided oracle.

Let us consider the value

$$d = (r^N \bmod N) + kN$$

Where  $k \leftarrow \mathbb{Z}_N$ . Note that, since  $r^N \bmod N$  is uniformly distributed in  $\mathbb{Z}_N^*$  and, being  $\mathbb{Z}_{N^2}^*$  isomorphic to  $\mathbb{Z}_N^* \times \mathbb{Z}_N$  [11],  $d$  is uniformly distributed in  $\mathbb{Z}_{N^2}^*$  and can be written (uniquely) as

$$d = r^N(1 + mN) \bmod N^2$$

extracting  $m$  from  $d$  (via the given oracle) thus leads to compute  $r^N \bmod N^2$ .  $\square$

*Remark 1.* At PKC'01 Damgård and Jurik [6] presented a generalized (and still homomorphic) version of Paillier's basic cryptosystem in which the expansion factor is reduced and the block length of the scheme may be changed without altering the public key. Moreover they show that such a variant is as secure as Paillier's construction.

The result presented in Theorem 2 above, can be generalized to connect the one-wayness of the Damgård-Jurik construction and the hardness of Hensel-RSA with appropriate parameters. Details are deferred to the final version of this paper.

## 4 The Discrete Log Case

In this section we extend our results to the discrete logarithm function. Let  $\omega \in \mathcal{PRIMES}(k)$  and  $g \in \mathbb{Z}_p^*$  an element of order  $\omega$  in  $\mathbb{Z}_p^*$ , where  $p$  is a prime (note that  $\omega$  must divide  $p - 1$ ). We introduce the following, computational, problem: Given  $p, g, \omega$  and  $h = g^x \bmod p$ , compute  $h' = g^x \bmod p^\ell$ .

Formally we define the function:

$$\mathbf{Hensel - Dlog}[p, g, \ell](g^x \bmod p) = g^x \bmod p^\ell$$

We will assume this function to be not computable in probabilistic polynomial time.

**Definition 3.** Let  $n(\cdot)$  be a polynomial, we say that computing the function  $\mathbf{Hensel-Dlog}[p, g, \ell](g^x \bmod p)$  is hard if, for every probabilistic polynomial time algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\cdot)$  such that

$$\Pr \left[ \begin{array}{l} \omega \leftarrow \mathcal{PRIMES}(k) \\ p \leftarrow \mathcal{PRIMES}(n(k)) \text{ s.t. } p - 1 \propto \omega \\ g \leftarrow \mathbb{Z}_p^* \text{ s.t. } \text{ord}(g) = \omega \\ x \leftarrow \mathbb{Z}_\omega; \ h = g^x \bmod p; \\ \mathcal{A}(N, g, h, \omega, \ell) = g^x \bmod p^\ell \end{array} \right] = \text{negl}(k)$$

With the following theorem we relate the hardness of the function Hensel-Dlog, to the hardness of the Discrete Logarithm function.

**Theorem 3.** Let  $\omega$  be a  $k$ -bit random prime and  $p$ , such that  $p - 1 \propto \omega$ , a prime whose size is polynomially related with  $k$ . Given  $g$  of order  $\omega$  in  $\mathbb{Z}_p^*$ ,  $p$  and  $\omega$ ,  $\mathbf{Hensel-Dlog}[p, g, \ell]$  is hard if and only if the discrete logarithm in the subgroup spanned by  $g$  in  $\mathbb{Z}_p^*$  is a one-way function, where  $\ell$  is defined as the unique positive integer such that  $g^\omega \not\equiv 1 \pmod{p^\ell}$  and  $g^\omega \equiv 1 \pmod{p^{\ell-1}}$ .

*Proof.* We follow the proof of Theorem 1. Assume, for the sake of contradiction, that  $\mathbf{Hensel-Dlog}[p, g, \ell]$  is not hard. This means that there exists an oracle  $\mathcal{O}$  that, on input a random challenge  $h = g^x \bmod p$  uniformly distributed over the subgroup spanned by  $g$ , computes  $g^x \bmod p^\ell$  with some non-negligible probability  $\varepsilon$ . Here we will show how to use this oracle to construct a probabilistic



polynomial time algorithm  $\mathcal{I}$  that succesfully extracts discrete logarithms in base  $g$  modulo  $p$  with a polynomially related probability.

We are given as input a random element  $h = g^x \bmod p$ : our goal is to compute  $x$ . We start by choosing a random  $a$  uniformly in  $\mathbb{Z}_\omega^*$ . We then call the oracle  $\mathcal{O}$  twice, on inputs  $h$  and  $h^a \bmod p$ . Since the queries  $h$  and  $h^a \bmod p$  are independent and uniformly distributed over the subgroup spanned by  $g$  (because  $\omega$  is prime), we obtain with probability  $\varepsilon^2$  the integers  $g^x \bmod p^\ell$  and  $g^\mu \bmod p^\ell$  where  $\mu$  is defined by  $\mu = ax \bmod \omega$ .

Because  $0 \leq a < \omega$  and  $0 \leq x < \omega$ , there exists an integer  $r$  such that  $ax = \mu + r\omega$  and  $0 \leq r < \omega$ . We obtain:

$$g^{ax} \equiv g^\mu g^{r\omega} \pmod{p^\ell}.$$

From  $g^x \bmod p^\ell$  and  $g^\mu \bmod p^\ell$ , we therefore derive  $g^{r\omega} \bmod p^\ell$ . Besides,  $\ell$  is such that  $g^\omega \not\equiv 1 \pmod{p^\ell}$  and  $g^\omega \equiv 1 \pmod{p^{\ell-1}}$ . One can therefore compute an integer  $z \in \mathbb{Z}_p$  such that:

$$g^\omega \equiv 1 + p^{\ell-1}z \pmod{p^\ell}.$$

Then:

$$g^{r\omega} \equiv 1 + p^{\ell-1}rz \pmod{p^\ell}.$$

Hence, we can compute  $r \bmod p$ , and since  $0 \leq r < \omega < p$ , we know  $r$  exactly.

Now, in the equation  $ax = \mu + r\omega$ , only the integers  $0 \leq x < \omega$  and  $0 \leq \mu < \omega$  are unknown. We have:

$$\frac{r\omega}{a} \leq x < \frac{(r+1)\omega}{a}.$$

We thus obtain an interval of length  $\omega/a$  containing  $x$ . We now show that with overwhelming probability, this interval will be sufficiently short to be exhaustively searched.

Indeed, with probability at least  $1 - \varepsilon^2/\log \omega$  over the choice of  $a$ , we have  $a \geq \varepsilon^2\omega/\log \omega$ , which implies that  $0 \leq \omega/a \leq (\log \omega)/\varepsilon^2$ . It follows that with probability at least  $\varepsilon^2 - \varepsilon^2/\log \omega = \varepsilon^2(1 - 1/\log \omega)$  over the choice of  $(r, a)$ , we have  $0 \leq \omega/a \leq (\log \omega)/\varepsilon^2$  and the outputs of the two oracle calls are correct. Then, by exhaustive search over at most  $(\log \omega)/\varepsilon^2 \leq k/\varepsilon^2$  possibilities, we obtain  $x$  (the correct value can be recognized by the congruence  $h \equiv g^x \pmod{p}$ ). Thus, with probability at least  $\varepsilon^2(1 - 1/\log \omega) = \varepsilon^2(1 - o(1))$  (as  $k$  grows), we can compute  $x$  in time polynomial in  $k$  and  $1/\varepsilon$ .  $\square$

## 5 Conclusions

In this paper we introduced two new functions and we studied their computational properties by relating them to the problems of inverting RSA and computing discrete logarithms. Moreover we formally proved that the one-wayness of the RSA-Paillier scheme [4] is actually equivalent to that of RSA, thus fixing an incorrect proof recently proposed by Sakurai and Takagi [12].

There are several open questions arising from this research. It would be nice to know whether it is possible to further extend our results to discover the exact relation existing between Paillier's Class assumption and  $\text{RSA}[N, N]$ . Another intriguing direction may be to try to improve our understanding about the hardness of the Hensel-Dlog function, and to find cryptographic applications. We proved that if one can compute  $g^x \bmod p^\ell$  from  $g^x \bmod p$  (in the case when  $g^\omega \equiv 1 \bmod p$  but  $g^\omega \not\equiv 1 \bmod p^\ell$ ) then one could compute the discrete logarithm function over the subgroup spanned by  $g$ . This implies that computing  $g^x \bmod p^{\ell-1}$  from  $g^x \bmod p$  may be potentially easier than computing discrete logarithms in the subgroup spanned by  $g$ .

## Acknowledgements

We would like to thank Igor Shparlinski for helpful discussions.

## References

1. E. Bach and J. Shallit. *Algorithmic Number Theory, Vol.1: Efficient Algorithms*. MIT Press, 1996.
2. M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, No. 4:850-864, 1984.
3. D. Catalano, R. Gennaro and N. Howgrave-Graham. The Bit Security of Paillier's Encryption Scheme and its Applications. In *Advances in Cryptology - Eurocrypt '01*. LNCS vol.2045, Springer, 2001, pages 229-243.
4. D. Catalano, R. Gennaro, N. Howgrave-Graham and P. Q. Nguyen. Paillier's Cryptosystem Revisited. In *8th ACM Conference on Computer and Communication Security* pp.206-214, 2001.
5. H. Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics, Vol 138, Springer, 1996.
6. I. Damgård and M. Jurik. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Public key Cryptography*, LNCS vol. 1992, 2001, pages 119-136.
7. R. Fischlin and C.P. Schnorr. Stronger Security Proofs for RSA and Rabin Bits. *J. of Cryptology*, 13(2):221-244, Spring 2000.
8. F. Gouvêa. *p-adic numbers*. Universitext, Springer, 1997.
9. M. Grötschel, L. Lovász and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993.
10. P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In *Proc. of CALC '01*, volume 2146 of LNCS, Springer-Verlag, 2001.
11. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - Eurocrypt '99*, LNCS vol. 1592, Springer, 1997, pages 223-238.
12. K. Sakurai and T. Takagi. New Semantically Secure Public-Key Cryptosystems from the RSA Primitive In *Public key Cryptography*, LNCS vol. 2274, 2002, pages 1-16.
13. T. Takagi. Fast RSA type Cryptosystems Using n-adic Expansion. In *Proc. of Crypto '97*, volume 1294 of LNCS, Springer-Verlag, 1997.

# A Comparison and a Combination of SST and AGM Algorithms for Counting Points of Elliptic Curves in Characteristic 2

Pierrick Gaudry

Laboratoire d'Informatique (CNRS/UMR 7650), École polytechnique,  
91128 Palaiseau Cedex, France,  
`gaudry@lix.polytechnique.fr`

**Abstract.** Since the first use of a  $p$ -adic method for counting points of elliptic curves, by Satoh in 1999, several variants of his algorithm have been proposed. In the current state, the AGM algorithm, proposed by Mestre is thought to be the fastest in practice, and the algorithm by Satoh–Skjernaa–Taguchi has the best asymptotic complexity but requires precomputations. We present an amelioration of the SST algorithm, borrowing ideas from the AGM. We make a precise comparison between this modified SST algorithm and the AGM, thus demonstrating that the former is faster by a significant factor, even for small cryptographic sizes.

## 1 Introduction

In the design of an elliptic public key cryptosystem, parameter initialization is a difficult task; it is required to count points of curves until one is found with almost prime group order. In the early ages of elliptic curve cryptography, the only way to achieve this was to use curves with special properties like having complex multiplication by a small discriminant or being supersingular, though taking random curves might be seen as the most secure. The first polynomial time algorithm for point-counting was designed in 1985 by Schoof [11], but was not fast enough to deal with cryptographic sizes. A decade of theoretical and practical improvements by Atkin, Couveignes, Dewaghe, Elkies, Lercier, Morain, Müller lead to a situation where point-counting was efficiently feasible for cryptographic sizes (see the survey [1] and the references therein). However, the cost of parameter initialization remained high (in runtime and in complexity of programming) compared to other systems like RSA or XTR. Situation changed in 1999 when Satoh [8] proposed a new algorithm for counting points of elliptic curves over finite field of small characteristic. His method is based on the computation of the canonical lift of the curve in a  $p$ -adic local ring. The theoretical complexity is asymptotic better than all the variants of Schoof's algorithm. Further work by Fouquet–Gaudry–Harley [23], Skjernaa [12], Vercauteren et al. [13], Satoh–Skjernaa–Taguchi [9,10], Hae Young Kim et al. [7] made this algorithm practical, in particular in characteristic 2, which is the most important in

practice. Another closely related method, based on the algebraic-geometric mean (AGM) was found by Mestre, and an implementation by Harley [5] proved it to be very efficient. For a curve over  $\mathbb{F}_{2^n}$ , the complexity of all these method is in  $O(n^{3+\varepsilon})$ , except for the Satoh–Skjernaa–Taguchi (SST) method which achieves a complexity in  $O(n^{2.5+\varepsilon})$  but requires precomputations.

In characteristic 2, with the current state of the art, AGM is thought to be the fastest for cryptographical sizes, due to a very small constant, and is also the best algorithm for computing records. The first drawback of the SST algorithm is the precomputation stage which is not feasible for records, but this is not at all a problem for cryptographical sizes, where this is easily doable and the storage of the precomputed data is manageable. Another problem in the SST method is that the defining polynomial for the local ring is dense, thus increasing the cost of a multiplication by a factor of 3 compared to the sparse structure used in AGM.

Our contribution is two-sided: firstly we mix ideas of SST and AGM to get what we call the modified SST algorithm (MSST) which is faster by a constant factor compared to the original SST algorithm. Our improvement modifies only the lifting phase; the norm computation which is common to both algorithm is not modified. Then we make a precise comparison between MSST and AGM algorithms, based on an evaluation of the number of operations required at each precision. This turns out to be in favor of MSST, even for small cryptographical sizes, as confirmed by some experiments we did on a Pentium III: for curves over  $\mathbb{F}_{2^{163}}$  we get a speed-up by a factor of 4.15 and for curves over  $\mathbb{F}_{2^{239}}$  the factor is 4.90.

The paper is organized as follows: in Section 2 we recall some basics and fix notations. In Section 3 and 4, we give a brief description of the original SST and AGM algorithms. Section 5 is devoted to the mix of SST and AGM algorithms. Section 6 contains a theoretical comparison between MSST and AGM methods, and Section 7 contains the numerical experiments.

## 2 General Setting and Notations

Let  $\mathbb{F}_q$  be a finite field of characteristic 2, and let  $n$  be such that  $q = 2^n$ . Let  $E$  be a non-supersingular elliptic curve over  $\mathbb{F}_q$ . For the purpose of point-counting, without loss of generality, and perhaps considering the quadratic twist of  $E$ , we can assume that  $E$  has an equation of the form  $y^2 + xy = x^3 + a_6$ . Then its  $j$ -invariant is given by  $j = a_6^{-1}$ . Denote by  $N$  the group order of  $E$ . The *trace* of  $E$  is defined by  $\text{Tr}(E) = q + 1 - N$  and Hasse's theorem states that  $|\text{Tr}(E)| \leq 2\sqrt{q}$ .

All the  $p$ -adic point-counting methods proceed in the same way: lift some data from  $\mathbb{F}_q$  to a  $p$ -adic local ring with enough precision, and deduce a  $p$ -adic approximation of the  $\text{Tr}(E)$  which might be enough to conclude due to Hasse's bound.

In our case, the 2-adic local ring we shall consider is the ring of integers of the degree  $n$  unramified extension of  $\mathbb{Q}_2$ . In the following we denote this ring by  $\mathbb{Z}_q$  (noted  $W(\mathbb{F}_q)$  in some papers, or simply  $R$  in [10]). Note that  $\mathbb{Z}_q$  has nothing

to do with  $\mathbb{Z}/q\mathbb{Z}$ , just like the ring of 2-adic integers  $\mathbb{Z}_2$  is not  $\mathbb{Z}/2\mathbb{Z}$ . The ring  $\mathbb{Z}_q$  is equipped with a cyclic  $\mathbb{Z}_2$ -automorphism of order  $n$  which reduces to the 2-nd power Frobenius automorphism of  $\mathbb{F}_q$ .

We give a constructive way to see the ring  $\mathbb{Z}_q$ . Let  $\bar{f}(t)$  be the irreducible polynomial of degree  $n$  over  $\mathbb{F}_2$  chosen to define  $\mathbb{F}_q = \mathbb{F}_2[t]/(\bar{f}(t))$ . Consider any monic polynomial  $f(t)$  with coefficients in  $\mathbb{Z}_2$  which reduces to  $\bar{f}(t)$  modulo 2. Then  $f(t)$  is irreducible and  $\mathbb{Z}_q$  can be defined by  $\mathbb{Z}_q = \mathbb{Z}_2[t]/(f(t))$ . Different choices for the polynomial  $f(t)$  lead to isomorphic rings. From an algorithmic point of view two strategies can be used: choosing a sparse  $f(t)$  with small coefficients speeds up the basic arithmetic, because the reduction modulo  $f(t)$  is almost for free; this is the representation used in the AGM algorithm. On the other hand, lifting  $f(t)$  in a careful way can give a representation in which the Frobenius substitution is efficiently computable, the price to pay is a dense polynomial  $f(t)$ , hence a non-negligible reduction modulo  $f(t)$ ; this is the representation used in the SST algorithm.

## 2.1 Notations in $\mathbb{Z}_q$

The ring  $\mathbb{Z}_q$  comes with the natural “reduction modulo 2” homomorphism onto  $\mathbb{F}_q$ . For  $x \in \mathbb{Z}_q$ , the notation  $x \bmod 2$  means the element of  $\mathbb{F}_q$  image of  $x$  by this homomorphism. The ring  $\mathbb{Z}_q$  also comes with a *valuation*. Let  $x$  and  $y$  be elements of  $\mathbb{Z}_q$ ; the valuation of  $(x - y)$  is high when  $x$  and  $y$  are close to each other. More precisely, the valuation of  $(x - y)$  is  $k$  if  $(x - y)$  is in  $2^k\mathbb{Z}_q$ ; then we write  $x \equiv y \bmod 2^k\mathbb{Z}_q$ , or simply  $x \equiv y \bmod 2^k$ .

In an algorithm, when we say “compute  $x := \dots \bmod 2^k$ ”, this means that the expression for computing  $x$  involves quantities which are known to precisions such that the result is known to precision at least  $k$ . The variable  $x$  is then assigned an element which is congruent to the result modulo  $2^k$ .

We use the same notation  $\sigma$  for *all* kinds of 2-nd power Frobenius action: ring/field automorphisms of  $\mathbb{F}_q$  and  $\mathbb{Z}_q$ , and also their coordinate-wise extensions to isogenies from an elliptic curve to its conjugate. There should be no confusion, because all the domain of these maps are distinct and all the reduction-diagrams involving two of these  $\sigma$  commute.

## 2.2 Canonical Lift of an Elliptic Curve

It is easy to find many curves over  $\mathbb{Z}_q$  whose equations reduce to the equation of  $E$  modulo 2. However, there is a canonical way to do it and keep information on the group order.

**Theorem 1 (Lubin–Serre–Tate).** *Let  $E$  be a non supersingular elliptic curve over  $\mathbb{F}_q$ . Then, up to isomorphism, there exists a unique curve  $\mathcal{E}$  defined over  $\mathbb{Z}_q$ , such that:*

1. *The equation of  $\mathcal{E}$  reduces to the equation of  $E$  modulo 2;*
2.  *$\text{End}(\mathcal{E}) \cong \text{End}(E)$ .*

A consequence of this theorem is the following commutative diagram

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{\sigma} & \mathcal{E}^\sigma \\ \downarrow & & \downarrow \\ E & \xrightarrow{\sigma} & E^\sigma \end{array}$$

where  $\sigma$  on the top arrow is the 2-nd power Frobenius isogeny from  $\mathcal{E}$  to  $\mathcal{E}^\sigma$ . This isogeny being of degree 2, the modular equation of degree 2 relates the  $j$ -invariants of the canonically lifted curves:

$$\Phi_2(j(\mathcal{E}), j(\mathcal{E}^\sigma)) = 0,$$

where  $\Phi_2(X, Y)$  is the symmetric bivariate polynomial

$$\begin{aligned} X^3 + Y^3 - X^2Y^2 + 1488(XY^2 + X^2Y) - 162000(X^2 + Y^2) \\ + 40773375XY + 8748000000(X + Y) - 157464000000000. \end{aligned}$$

**Notation:** Let  $c$  be an element of  $\mathbb{Z}_q$  and let  $\bar{c}$  be its reduction modulo 2. If  $\bar{c}$  is not the  $j$ -invariant of a supersingular curve, then we denote by  $c^\uparrow$  the  $j$ -invariant of the canonical lift of an elliptic curve whose invariant is  $\bar{c}$ .

### 3 Satoh–Skjernaa–Taguchi Algorithm

We do not describe all the details of the SST algorithm: we concentrate on the main two steps: firstly the  $j$ -invariant of the canonical lift of  $E$  is computed to some precision, then the norm of a quantity is computed, yielding the trace. We refer to the original paper [10] for a description of the missing steps.

#### 3.1 Canonical Lifting of the $j$ -Invariant

We are given  $j(E)$  in  $\mathbb{F}_q \setminus \mathbb{F}_4$ , and we want to compute  $j(\mathcal{E})$ . The value of  $j(\mathcal{E})$  is determined one bit after the other: assume that we know  $J$  such that  $J \equiv j(\mathcal{E}) \pmod{2^k}$ , then writing  $j(\mathcal{E}) = J + 2^k e$ , and plugging it formally into the equation  $\Phi_2(j(\mathcal{E}), j(\mathcal{E}^\sigma)) = 0$ , one gets an equation yielding  $e$  modulo 2. Hence we have gained one bit on the approximation of  $j(\mathcal{E})$ .

For a more precise setting, we take a Taylor expansion:

$$\begin{aligned} 0 &= \Phi_2(j(\mathcal{E}), j(\mathcal{E}^\sigma)) = \Phi_2(J + 2^k e, J^\sigma + 2^k e^\sigma) \\ &= \Phi_2(J, J^\sigma) + 2^k e \partial_X \Phi_2(J, J^\sigma) + 2^k e^\sigma \partial_Y \Phi_2(J, J^\sigma) + 2^{2k-1} (\text{element of } \mathbb{Z}_q). \end{aligned}$$

In this equation,  $\Phi_2(J, J^\sigma)$  is zero modulo  $2^k$ , we can therefore divide everything by  $2^k$ . Furthermore the Kronecker relation implies that  $\partial_X \Phi_2(J, J^\sigma)$  is zero modulo 2 and that  $\partial_Y \Phi_2(J, J^\sigma)$  is different from zero modulo 2. Finally we have  $e^\sigma \equiv e^2 \pmod{2}$  and we get

$$e^2 \equiv \frac{\Phi_2(J, J^\sigma)}{2^k \partial_Y \Phi_2(J, J^\sigma)} \pmod{2}.$$

**Algorithm 1.***Input:*  $j$  and a desired precision  $k$ .*Output:*  $j^\dagger$  to precision  $k$ .

1.  $d := 1/\partial_Y \Phi_2(\sigma^{-1}(j), j)$ ;
2.  $y := j$ ;
3. For  $i$  from 1 to  $k - 1$  do
4.    $x := \sigma^{-1}(y) \bmod 2^{i+1}$ ;
5.    $y := y - d\Phi_2(x, y) \bmod 2^{i+1}$ ;
6. Return  $y$ ;

To turn this into an algorithm, we need to apply  $\sigma$  to elements of  $\mathbb{Z}_q$ . For this, the polynomial defining  $\mathbb{Z}_q$  over  $\mathbb{Z}_2$  is chosen such that its roots are  $(q - 1)$ -th roots of unity. This polynomial is precomputed once for all, for each base field. Hence Frobenius substitution is just reordering the coefficients and reducing modulo the defining polynomial (see [9] for details).

Then we get the following first lifting algorithm: (inverse of Frobenius is used, thus saving the computation of the square root of  $e$  in the previous formula)

In a point counting context, we need  $k \approx \frac{n}{2}$ . Running the  $i$ -th loop requires 1 Frobenius substitution and  $O(1)$  multiplications of elements of  $\mathbb{Z}_q$  at precision  $2^{i+1}$ . Therefore the cost of Algorithm 1 is in  $O(n^{1+2\mu})$  bit-operations, which is the cost of other lifting methods. Here  $\mu$  is a real such that multiplication of  $k$  bit objects can be done in time  $O(k^\mu)$ .

When looking at what is going on in Step 5, we see that  $\Phi_2(x, y)$  is very close to zero and only the small non-zero piece of information is used to update  $y$ . It looks sub-optimal to recompute all the time  $\Phi_2(x, y)$  from scratch: the values of  $x$  and  $y$  at step  $i + 1$  are close to the ones at step  $i$ , therefore  $\Phi_2(x, y)$  at step  $i + 1$  can be deduced from its value at step  $i$  and some adjustment involving partial derivatives. By precomputing the partial derivatives modulo  $2^W$ , one can update  $\Phi_2(x, y)$  during  $W$  iterations, then one needs to recompute one time  $\Phi_2(x, y)$  from scratch before doing again  $W$  iterations with only cheap updates. These ideas yield the SST lifting algorithm, a sketch of which is reproduced in Algorithm 2.

In [10] it is shown that Algorithm 2 runs in time  $O(n^{2\mu+1/(\mu+1)})$ , when one chooses  $W = n^{\mu/(\mu+1)}$ . Therefore, it is always better than Algorithm 1, because  $\mu > 1$ . If an FFT-based multiplication algorithm is used,  $\mu = 1 + \varepsilon$ , and we get a complexity of  $O(n^{2.5+\varepsilon})$ .

From the lifted  $j$ , a quantity can be derived which is a rational fraction in  $j$ , such that the norm of this quantity gives the trace of  $E$ .

### 3.2 Fast Norm Computation

In [10] a fast norm computation is described, that is well-suited to the case of point-counting in characteristic 2. It is based on the following equation:

**Algorithm 2. SST canonical lifting***Input:*  $j$  and a desired precision  $k$ ; a parameter  $W$ .*Output:*  $j^\dagger$  to precision  $k$ .

1.  $y := j^\dagger \bmod 2^W$ , computed via Algorithm 1;
2.  $D_X := \partial_X \Phi_2(\sigma^{-1}(y), y) \bmod 2^W$ ;  
 $D_Y := \partial_Y \Phi_2(\sigma^{-1}(y), y) \bmod 2^W$ ;
3. For  $m$  from 1 to  $\lfloor \frac{k-1}{W} \rfloor$  do
4.   Lift arbitrarily  $y$  modulo  $2^{(m+1)W}$ ;
5.    $V := \Phi_2(\sigma^{-1}(y), y) \bmod 2^{(m+1)W}$ ;
6.   For  $i$  from 0 to  $W-1$  do
7.     Compute  $y := j^\dagger \bmod 2^{mW+i+1}$ ;
8.     Update  $V := \Phi_2(\sigma^{-1}(y), y) \bmod 2^{mW+i+1}$ ;  
       // Steps (7) and (8) use only operations modulo  $2^W$ ;
9. Return  $y$ ;

$$\text{Norm}(x) = \exp(\text{Tr}(\log(x))),$$

which holds whenever  $\log$  and  $\exp$  converge. Computing a trace is far easier than computing a norm, and the subsequent exponential is very cheap. Therefore the main cost is a log evaluation, which is performed by fast evaluation of power series. We refer to the original paper for details, and will not discuss this part anymore, since the norm computation is a step which has to be done for any variant of the algorithm, and it is not the place where one is better than the other.

## 4 AGM Algorithm

We recall here the principles of the AGM algorithm. We give no proof of the results, they can be derived in the similar manner as for the other point-counting algorithms and are out of the scope of this paper.

### 4.1 The Arithmetic-Geometric Mean (AGM) Sequence

Let  $a_6$  be an element of  $\mathbb{F}_q^*$  and let  $E$  be the curve of equation  $y^2 + xy = x^3 + a_6$  with  $j$ -invariant  $j(E) = a_6^{-1}$ . We denote also by  $a_6$  an arbitrary element of  $\mathbb{Z}_q$  that reduces to  $a_6$  modulo 2. We then have recursive formulae which give a well-defined sequence  $(A_i, B_i)$  of elements of  $\mathbb{Z}_q$ :

$$A_0 = 1 + 8a_6, \quad B_0 = 1,$$

$$A_{i+1} = \frac{A_i + B_i}{2}, \quad B_{i+1} = \sqrt{A_i B_i},$$

where the square root is chosen to be congruent to 1 modulo 4. Indeed, by induction, we show that if  $A_i \equiv B_i \equiv 1 \pmod{4}$  and  $\frac{A_i}{B_i} = 1 + 8\alpha$  for an invertible



$\alpha$ , then the squareroot is possible to be taken and if the one which is chosen is congruent to 1 modulo 4, the same properties hold at step  $i + 1$ .

This sequence  $(A_i, B_i)$  is called the *AGM sequence* and we can associate to it the sequence of elliptic curves  $E_i$  of equations  $y^2 = x(x - A_i^2)(x - B_i^2)$ . We denote by  $j_i$  the  $j$ -invariant of the curve  $E_i$ .

## 4.2 Link with Canonical Lifting

It is well known that AGM is linked to isogenies of degree 2 between elliptic curves. This in turn gives a link with the canonical lifting as follows:

**Theorem 2.** *Let  $j_i$  be the  $j$ -invariant of the curve  $E_i$  attached to the AGM sequence. Then the sequence  $j_i$  verifies:*

$$j_0 \equiv a_6^{-2} \pmod{2},$$

$$j_{i+1} \equiv j_i^2 \pmod{2},$$

$$j_i \equiv j_i^\uparrow \pmod{2^{i+2}}.$$

The first assertion shows that  $E_0$  is isomorphic to the conjugate of a lift of the initial curve  $E$  modulo 2. The second relation states that all the curves  $E_i$  also reduce modulo 2 to conjugates of the curve  $E$  (up to isomorphism). The third one is the heart of the AGM algorithm: it means that when progressing along the AGM sequences, we get closer and closer to the canonical lift.

This yields immediately a straightforward algorithm for computing the canonical lift. Starting with the initial values of  $(A_0, B_0)$ , we apply the recursive formula to compute successive values of  $(A_i, B_i)$ . After  $k$  steps, we can compute the  $j$ -invariant of the associated curve which is close to the canonical lifting of a conjugate of  $E$  up to precision about  $2^k$ .

The link with the trace is given by the following result:

**Theorem 3.** *Let  $i > 0$  and let  $c_i$  be  $\text{Norm}_{\mathbb{Z}_q/\mathbb{Z}_p} \left( \frac{A_{i+1}}{A_i} \right)$ . Then*

$$c_i + \frac{q}{c_i} \equiv \text{Tr}(E) \pmod{2^{i+4}}.$$

A point-counting algorithm follows easily: one computes the AGM sequence with enough steps, then a norm computation gives the trace of the initial curve up to some precision which is equal to the number of steps plus a constant. A practical complication arises: on a computer one cannot really deal with elements of  $\mathbb{Z}_q$ , but with truncated ones. At first sight, it seems that we need a high starting precision for  $A_0$  and  $B_0$ , because we get less and less significant digits on  $A_i$  and  $B_i$  when at the same time the  $j_i$  gets closer to the canonical lift. This problem can be overturned by adding arbitrary noise to  $A_i$  and  $B_i$  just before doing an operation which “loses” precision like a square root or a division by 2.

After having cleaned the details we get the following algorithm:

**Algorithm 3. AGM point-counting***Input:*  $a_6$  in  $\mathbb{F}_{2^n}^*$ .*Output:* Trace of the curve  $y^2 + xy = x^3 + a_6$ .

1.  $a := 1 + 8a_6 \bmod 16$ ;  $b := 1 \bmod 16$ ;  $k := 4$ ;
2. Repeat until  $k = \lceil \frac{n}{2} \rceil + 3$ :
3. Lift arbitrarily  $a$  and  $b$  modulo  $2^{k+2}$ ;
4.  $(a, b) := \left( \frac{a+b}{2} \bmod 2^{k+1}, \sqrt{ab} \bmod 2^{k+1} \right)$ ;
5.  $k := k + 1$ ;
6.  $a' := \frac{a+b}{2}$ ;
7. Return  $\text{Norm}\left(\frac{a'}{a}\right) \bmod 2^{\lceil \frac{n}{2} \rceil + 2}$  as a signed integer in  $[-2\sqrt{2^n}, 2\sqrt{2^n}]$ .

**4.3 Runtime Analysis**

The AGM algorithm requires  $O(n)$  operations between elements of  $\mathbb{Z}_q$  with maximal precision in  $O(n)$ , and then a norm computation. As said before, the norm computation will not be discussed here, because it is the same in every algorithm. The cost of the lifting process is in  $O(n^{1+2\mu})$ , which is the same as Algorithm 1. Hence Algorithm 2 by Satoh–Skjerna–Taguchi is asymptotically faster than the AGM for the lifting phase (but requires precomputation). However the AGM algorithm has a very low constant, due to the small number of operations at each step and the fact that a sparse defining polynomial for  $\mathbb{Z}_q$  can be used. The figures given in [10] suggest that for cryptographical sizes, AGM remains faster.

**5 AGM-Aided SST Algorithm**

The AGM algorithm as stated before does not appear to be mixable with the SST idea. Therefore, before doing so we need to rewrite it in a univariate way to reveal the hidden modular equation which can then be used instead of  $\Phi_2$  in the SST algorithm. This is also this version of the AGM algorithm that we shall use for the comparison and the implementation in the next sections. We do not expect any speed difference between the univariate and the bivariate AGM.

**5.1 Univariate AGM Algorithm**

Taking again the AGM sequence as a starting point, we define a new sequence

$$\lambda_i = \frac{A_i}{B_i}.$$

The corresponding curves have equation  $y^2 = x(x-1)(x-\lambda_i^2)$ . An easy computation shows that  $\lambda_{i+1}$  can be computed directly from  $\lambda_i$  by

$$\lambda_{i+1} = \frac{1 + \lambda_i}{2\sqrt{\lambda_i}}.$$

Another important fact is that  $\lambda_{i+1} \equiv \lambda_i^\sigma \pmod{2^{i+4}}$ , which corresponds to the fact that we are jumping from a curve to an approximation of its conjugate.

The corresponding univariate AGM algorithm is as follows:

**Algorithm 4. Univariate AGM**

*Input:*  $a_6$  in  $\mathbb{F}_{2^n}^*$ .

*Output:* Trace of the curve  $y^2 + xy = x^3 + a_6$ .

1.  $\lambda := 1 + 8a_6 \pmod{16}$ ;  $k := 4$ ;
2. Repeat until  $k$  is  $\lceil \frac{n}{3} \rceil + 3$ :
3. Lift arbitrarily  $\lambda$  modulo  $2^{k+2}$ ;
4.  $\lambda := \frac{1+\lambda}{2\sqrt{\lambda}} \pmod{2^{k+1}}$ ;
5.  $k := k + 1$ ;
6. Return  $\text{Norm}(\frac{2\lambda}{1+\lambda}) \pmod{2^{\lceil \frac{n}{2} \rceil + 2}}$  as a signed integer in  $[-2\sqrt{2^n}, 2\sqrt{2^n}]$ .

**Implementation of the Square Root.** The main step of this algorithm is Step (4) in which we have to compute the inverse of the square root of  $\lambda$  and to multiply the result by  $\frac{1+\lambda}{2}$ . The inverse of the square root is computed via a Newton iteration that can be done without inversion. First we note that  $\lambda$  is always of the form  $\lambda \equiv 1 + 8\alpha \pmod{16}$ . Then  $\frac{1}{\sqrt{\lambda}} \equiv 1 - 4\alpha \pmod{8}$ , and this will be the initialization of the lift. Then the iteration is

$$x_{n+1} := x_n + \frac{x_n}{2}(1 - \lambda x_n^2).$$

If for some  $n$ ,  $x_n$  is congruent to  $\frac{1}{\sqrt{\lambda}} \pmod{2^k}$ , then one can show that  $x_{n+1}$  is equal to  $\frac{1}{\sqrt{\lambda}} \pmod{2^{2k-1}}$  (see for instance Lemma 2.7 in [2]).

## 5.2 Modified Modular Equation

In the previous algorithm, the  $\lambda_i$  which is computed at the last step of the loop is (the conjugate of) a solution of the following equations:

$$Z \equiv 1 + 8a_6 \pmod{16},$$

$$(Z^\sigma)^2(1 + Z)^2 - 4Z \equiv 0 \pmod{2^i}.$$

But this is precisely this kind of system that SST algorithm is meant to solve. It remains to remove the leading non-significant bits in  $\lambda_i$  and to prove the same result on partial derivatives that made Algorithm 1 work.

Let  $E(X, Y)$  be the AGM modular equation:

$$E(X, Y) = Y^2(1 + X)^2 - 4X = 0.$$

We make a change of variables  $X \leftarrow 1 + 8X$ ,  $Y \leftarrow 1 + 8Y$ , and the modular equation becomes:

$$\tilde{E}(X, Y) = (X + 2Y + 8XY)^2 + Y + 4XY = 0,$$

which has to be solved, subject to the conditions that  $X$  is known and non-zero modulo 2 and  $Y = X^\sigma$ .

The partial derivatives of  $\tilde{E}$  evaluated at  $(X, Y)$  give

$$\begin{aligned}\partial_X \tilde{E}(X, Y) &= 2(X + 2Y + 8XY)(1 + 8Y) + 4Y \\ \partial_Y \tilde{E}(X, Y) &= (1 + 4X)(1 + 4(X + 2Y + 8XY))\end{aligned}$$

This proves that  $\partial_X \tilde{E}(X, Y)$  is congruent to 0 modulo 2, whereas  $\partial_Y \tilde{E}(X, Y)$  is 1 modulo 2, thus yielding the required asymmetry for the SST algorithm to converge. Note also that the partial derivative with respect to  $Y$  is 1 modulo 2, so that it is no longer necessary to compute  $d$  in Step (1) of Algorithm [III](#).

### 5.3 Modified Satoh–Skjernaa–Taguchi (MSST) Algorithm

According to the previous section, it is possible to use SST algorithm to compute the lifted invariant of the curve (or more precisely some kind of Legendre’s invariant of the canonical lift of the curve). It remains to compute the data whose norm will give the result. This is actually much simpler than in the original SST algorithm: transposing the results of the AGM method, we can see that if  $\lambda$  is a solution of  $\tilde{E}(X, X^\sigma)$  and  $\lambda \equiv a_6 \pmod{2}$  then the following holds:

$$\text{Tr}(E) \equiv \text{Norm} \left( \frac{1}{1 + 4\lambda} \right) \pmod{2^n}.$$

We obtain Algorithm [5](#).

The advantage of the MSST algorithm is 2-sided: firstly the modular equation is smaller thus reducing by a constant factor the number of operations, secondly the intermediate step between the lift and the norm does not exist any more, thus simplifying the code and giving a slight speed-up.

## 6 Theoretical Comparison

The MSST algorithm is always faster than the plain SST algorithm because it involves strictly less operations. It remains to compare it to the AGM algorithm.

### 6.1 Constraint Environments

In a constraint environment it might be preferable to choose an algorithm for which no precomputation need to be stored and the RAM requirement stays low. In this context, the AGM algorithm (with the norm computation replaced by an extra loop) is by far the best choice. However, one should keep in mind that

**Algorithm 5. MSST point-counting***Input:*  $a_6$  in  $\mathbb{F}_{2^n}^\times$ .*Output:* Trace of the curve  $y^2 + xy = x^3 + a_6$ .

1.  $y := a_6$ ; // arbitrary lift to  $2^W$ .
2. For  $i$  from 1 to  $W - 1$  do
3.    $x := \sigma^{-1}(y) \bmod 2^{i+1}$ ;
4.    $y := y - \tilde{E}(x, y) \bmod 2^{i+1}$ ;
5.    $x := \sigma^{-1}(y) \bmod 2^W$ ;
6.    $D_X := \partial_X \tilde{E}(x, y) \bmod 2^W$ ;    $D_Y := \partial_Y \tilde{E}(x, y) \bmod 2^W$ ;
7. For  $m$  from 1 to  $\lceil \frac{n}{2W} \rceil$  do
8.   Lift arbitrarily  $y$  modulo  $2^{(m+1)W}$ ;
9.    $x := \sigma^{-1}(y) \bmod 2^{(m+1)W}$ ;
10.    $V := \tilde{E}(x, y) \bmod 2^{(m+1)W}$ ;
11.   For  $i$  from 0 to  $W - 1$  do   // break if  $i + mW \geq \lceil \frac{n}{2} \rceil$
12.      $\Delta_Y := -2^{-mW} V \bmod 2^W$ ;
13.      $\Delta_X := \sigma^{-1}(\Delta_Y) \bmod 2^W$ ;
14.      $y := y + 2^{mW} \Delta_Y \bmod 2^{(m+1)W}$ ;
15.      $V := V + 2^{mW} (D_X \Delta_X + D_Y \Delta_Y) \bmod 2^{(m+1)W}$ ;
16. Return  $\text{Norm}(\frac{1}{1+4y}) \bmod 2^{\lceil \frac{n}{2} \rceil + 2}$  as a signed integer in  $[-2\sqrt{2^n}, 2\sqrt{2^n}]$ .

for a reasonable key-size, the amount of precomputed data to be stored for the SST algorithm is not so high: this is essentially two elements of  $\mathbb{Z}_q$  at maximal precision; for instance, for  $\mathbb{F}_{2^{163}}$ , it is only a few kilo-bytes. This might be too much for a smart card, but this is not a problem on a PDA.

We consider as unlikely that someone really wants to count points in a highly constraint environment. Indeed, this kind of computation is required during the setup of the system parameters and the result does not need at all to be secret. Hence a card can ask to the server to do the computation if a new parameter setting is required.

In the following, we shall therefore concentrate on the case where we have no constraints and a machine word size of 32 bits (still the most common for PC's).

## 6.2 Assumptions

We recall that we are interested in cryptographically useful sizes. In that case, it has been shown in [10] that in the SST algorithm it is not worthwhile to use a  $W$  parameter different from the machine word size. Therefore we shall always consider that  $W = 32$  is the optimal parameter for the MSST algorithm.

Another assumption we make is that multiplying integers of size less than the machine word size is not significantly faster than multiplying integers of size exactly the machine word size. At first sight, this assumption looks reasonable since the assembly instructions for multiplying bytes or short integers usually do not require much less cycles than the instruction for long integers. In fact this

is a bit misleading because in both AGM and MSST algorithms several of these operations are parallelizable, one could therefore pack several small integers in a word size integer and perform several multiplication at once, or one could also use specific multimedia instruction like the MMX or SSE2 instruction set in the case of the Pentium. Using those could speed-up both the algorithms and we shall consider that we do not penalize one or the other by always using machine-word-size arithmetic.

### 6.3 Cost Analysis

In this section we compare the lifting parts of the AGM and the MSST algorithm. In MSST, we use a dense defining polynomial for  $\mathbb{Z}_q$  in order to speed-up the Frobenius substitution computation. Therefore the reduction modulo the defining polynomial subsequent to a multiplication or a square is costly. On the other hand, in the AGM we have no Frobenius substitution to perform and it is possible to use a sparse defining polynomial, leading to an almost free reduction. Therefore, reductions will be counted in the MSST algorithm whereas we shall neglect them in the AGM. In MSST, with the dense polynomial, Frobenius substitution can be done at a cost of roughly one multiplication and one reduction, as explained in [9]. We do not count additions and other simpler operations.

We use the following notations for the basic operations in  $\mathbb{Z}_q$ :

$P$  : unreduced product  
 $S$  : unreduced square  
 $R$  : reduction modulo defining polynomial

Furthermore, we can add an index  $i$  to each of these symbols to indicate the number of W-bit-digits of each operand.

**MSST Algorithm.** The cost to evaluate the modular equation  $\tilde{E}$  is  $P+S+2R$ . Each of its partial derivatives can be evaluated at a cost of  $2P+2R$ , and if one wants both derivatives, one can share the result of one product and get a cost of  $3P+3R$ .

Steps (2)-(4) cost  $(W-1)(2P_1+S_1+3R_1)$ .

Steps (5)-(6) cost  $4P_1+4R_1$ .

Next we analyze the cost of Steps (8)-(15) for each value of  $m$ :

Steps (9)-(10) cost  $2P_{m+1}+S_{m+1}+3R_{m+1}$ .

Steps (12)-(15) cost  $W(3P_1+2R_1)$ . Indeed, the multiplications by powers of 2 are for free, and one reduction can be saved at step (15). Actually, as explained in [10], some operations could be done on one bit operands.

If we need to lift to precision  $k$ , the total cost of MSST is then

$$(3k-W+2)P_1+(2k+W+1)R_1+(W-1)S_1+\sum_{1 \leq m \leq \lceil \frac{k-W}{W} \rceil} (2P_{m+1}+S_{m+1}+3R_{m+1}).$$

We then apply this formula with  $W = 32$ , for two base fields.

For  $\mathbb{F}_{2^{163}}$ , we need a precision  $k = 82$ , and we get

$$C_{MSST}(163) = 216P_1 + 31S_1 + 197R_1 + 2P_2 + S_2 + 3R_2 + 2P_3 + S_3 + 3R_3.$$

For  $\mathbb{F}_{2^{239}}$ , we need a precision  $k = 120$ , and we get

$$C_{MSST}(239) = 330P_1 + 31S_1 + 273R_1 + 2P_2 + S_2 + 3R_2 + 2P_3 + S_3 + 3R_3 \\ + 2P_4 + S_4 + R_4.$$

Hence we readily notice that for cryptographical sizes, only few operations require multiprecision arithmetic.

**AGM Algorithm.** As said above, we study the univariate AGM instead of the bivariate one. The key step is then clearly Step (4), and in this step the crucial part is the Newton iteration for computing the inverse of  $\sqrt{\lambda}$ :

$$x_{n+1} := x_n + \frac{x_n}{2}(1 - \lambda x_n^2).$$

As usual for a Newton iteration, we need to have operations with variable precision. If we want to compute  $x_{n+1}$  with precision  $k$  from  $x_n$  known at precision roughly  $k/2$ , at first sight it requires one square and two products computed modulo  $2^k$ . However this can be improved:  $(1 - \lambda x_n^2)$  is zero at precision  $k/2$ , because  $x_n$  is already a good approximation of the result. Hence when we multiply this further by  $x_n$ , this is actually a multiplication at precision  $k/2$  that has to be performed. One step further is explained by Karp and Markstein in [6]: the “self-correctingness” of this iteration allows to compute  $\lambda x_n^2$  with two multiplications of an operand at precision  $k$  and the other at precision  $k/2$ .

We note however that in our case a square ideally costs roughly one half of a product and that this “Full times Half” precision product saves one fourth of the operations. Thus the trick of Karp and Markstein does not help in our case.

Estimating the number of operations in a Newton iteration is not that easy, due to the variable precision. To simplify the formulae we shall consider that the precision is exactly doubled at each step (whereas in fact one bit is lost). On the other hand this is easy to write a short program that emulate the algorithm and count the number of operations and the precision required, because there is no branching depending on the input data. Hence for a given base field, it is much simpler to run this emulation to evaluate the cost. We shall compare the results given by both approaches.

*Cost of one lift in single precision.* The cost of the Newton iteration to get the inverse of the square root at a precision between  $2^{k-1}$  and  $2^k < W$  is  $(k-1)(2P_1 + S_1)$ . After the lift one has to multiply the result by  $\frac{1+\lambda}{2}$ , which cost another  $P_1$ .

*Cost of the first  $W$  iterations.* We split the interval  $[0, W]$  into pieces where the cost of the iterations are the same, and add them all. We get the following formula for the cost:

$$\sum_{2 \leq k \leq \log_2(W)} (k-1)2^{k-1}(2P_1 + S_1) + WP_1,$$

which simplifies to

$$(2 + W(\log_2(W) - 2))(2P_1 + S_1) + WP_1.$$

*Cost of the  $W$  iterations between precision  $W$  and  $2W$ .* The last step in a Newton iteration is done at precision of two  $W$ -bit digits and all the others are at precision 1. Furthermore, for each lift, the number of iteration at precision 1 is always  $(\log_2(W) - 1)(2P_1 + S_1)$ . Hence the cost of all the second  $W$  operations is

$$W\left((\log_2(W) - 1)(2P_1 + S_1) + P_1 + 2P_2 + S_2\right).$$

In this formula, we took into account the fact that in the last iteration at precision 2, one operation has only to be done at precision 1.

*Cost of the  $W$  iterations between precision  $2W$  and  $3W$ .* The last step in a Newton lift is done at a precision of 3 digits, thus reducing to compute the result at a precision between  $W$  and  $1.5W$ . The penultimate step is therefore at a precision of 2 digits, and the remaining steps are done at precision 1. The overall cost is then

$$W\left((\log_2(W) - 1)(2P_1 + S_1) + P_1 + 2P_2 + S_2 + 2P_3 + S_3\right).$$

*Cost of the  $W$  iterations between precision  $3W$  and  $4W$ .* The last step in a Newton lift is done at a precision of 4 digits, and the penultimate is done at a precision of 2 digits. The others steps are done at precision 1. Hence the following overall cost:

$$W\left((\log_2(W) - 1)(2P_1 + S_1) + P_1 + 2P_2 + S_2 + 2P_4 + S_4\right).$$

*Cost of one iteration at a precision of  $k$  digits.* The Newton iteration starts as always by  $(\log_2(W) - 1)$  operations at a precision of 1 digit. Then there are  $O(\log_2(k))$  operations at a precision of at most  $k$  digits. Including the cost of the multiplication by  $\frac{1+\lambda}{2}$ , there is at least  $2P_k + S_k$ .

We apply now our analysis to the same cases than for the MSST algorithm, namely  $n = 163$  and  $n = 239$ , with  $W = 32$ .

We get

$$C_{AGM,th}(163) = 696P_1 + 306S_1 + 104P_2 + 52S_2 + 40P_3 + 20S_3,$$

and

$$C_{AGM,th}(239) = 1038P_1 + 458S_1 + 180P_2 + 90S_2 + 64P_3 + 32S_3 + 52P_4 + 26S_4.$$

With the emulation program mentioned above we get:

$$C_{AGM,emu}(163) = 694P_1 + 303S_1 + 109P_2 + 57S_2 + 45P_3 + 23S_3,$$

and

$$C_{AGM,emu}(239) = 1033P_1 + 452S_1 + 188P_2 + 98S_2 + 64P_3 + 32S_3 + 57P_4 + 29S_4.$$

These values are close enough to justify the simplifications we made in our analysis.



## 6.4 Comparison

We first compare the cost of the lifting up to precision  $W = 32$ , where all the operations that take place are single precision. We have to compare

$$62P_1 + 31S_1 + 93R_1 \quad \text{and} \quad 228P_1 + 98S_1.$$

This clearly depends on the relative costs of  $R_1$  and  $P_1$ . It is always possible to do a reduction at the cost of two products, once a small precomputation is done (see [14], page 247). Hence  $R_1 \leq 2P_1$ . Note that in our implementation (see below) we got a ratio close to 1.5. Also  $S_1$  is usually about 1.5 faster than  $P_1$ . With these ratios, the advantage is on MSST side.

Next, we compare the costs for gaining  $W$  bits of precision at a higher level. This corresponds to one iteration of Steps (8)-(15) in MSST or  $W$  loops of AGM. Let  $k$  be the number of digits corresponding to the precision. In MSST, the cost is

$$2P_k + S_k + 3R_k + 96P_1 + 64R_1,$$

whereas in AGM we have

$$64P_k + 32S_k + \cdots + 256P_1 + 128S_1,$$

where the dots contain operations at a precision strictly between 1 and  $k$  digits.

Hence MSST is clearly faster than than AGM, and the difference increases with the size of the basefield, due to the higher number of operations at multi-precision in AGM.

## 7 Practical Experiments

We implemented the MSST and the univariate AGM algorithm in the C programming language, using the GNU MP library [4] for the low-level integer multiplications. Multiplications in  $\mathbb{Z}_q$  are done via Karatsuba algorithm. We wrote specific code for the machine word-size precision because in that case many things are simplified and this is critical in both algorithms. We give timings for two field sizes:  $n = 163$  and  $n = 239$ . All the experiments are made on a Pentium III at 700 MHz running Linux. The compiler is gcc version 2.96.

Field size	Precision	Product	Square	Reduction
163	1 word	0.11 <i>ms</i>	0.07 <i>ms</i>	0.14 <i>ms</i>
	2 words	1.4 <i>ms</i>	0.92 <i>ms</i>	2.3 <i>ms</i>
	3 words	1.8 <i>ms</i>	1.3 <i>ms</i>	3.6 <i>ms</i>
239	1 word	0.21 <i>ms</i>	0.13 <i>ms</i>	0.29 <i>ms</i>
	2 words	2.5 <i>ms</i>	1.7 <i>ms</i>	4.9 <i>ms</i>
	3 words	3.4 <i>ms</i>	2.3 <i>ms</i>	6.8 <i>ms</i>
	4 words	5.4 <i>ms</i>	4.5 <i>ms</i>	10.8 <i>ms</i>

Field size	Lift MSST	Lift AGM	Norm computation	Total MSST	Total AGM
163	0.08 s	0.49 s	0.05 s	0.13 s	0.54 s
239	0.26 s	1.82 s	0.14 s	0.40 s	1.96 s

We see that at low precision, we were able to get a reduction step faster than two times a product. For the two given field sizes, the ratio between the two lifted methods is a factor of 6 to 7 in favor of the MSST algorithm. We implemented the norm computation to get significant runtimes for the complete point-counting computation. However this part is not as well optimized as the first step and the runtime we give might be improved. We only mention that in case of MSST algorithm, after the lift, we switch to the sparse representation before calling the same norm routine as the one used for AGM. This base conversion can be made very quick by precomputing the corresponding matrix. For cryptographical sizes, this matrix fits easily in a few mega-bytes, but for records, this strategy is not feasible. For the complete computation, the overall gain we have by choosing MSST instead of AGM is respectively by a factor of 4.15 and 4.90 for fields of 163 and 239 bits.

For comparison, we recall that the runtimes given in [10] for the original SST algorithm were 0.76s for a field of 163 bits and 2.54s for 239 bits on a 866 MHz Pentium III.

## 8 Conclusion

We presented a modification of the SST algorithm, using ideas taken from the AGM algorithm to speed-up the lifting phase and remove the second phase; the norm computation is unchanged. We did a precise theoretical and experimental comparison between our method and the AGM. We demonstrate that the number of operations is much smaller for the former. To illustrate this we implemented both methods with the same level of optimization – actually, they use the same time-critical functions. The gain is significant, even for small cryptographical field sizes.

For cryptographical applications, it is required to have an almost prime group order. Therefore it is usually necessary to count  $O(\log(n))$  curves before finding one suitable for cryptography. To speed-up this search, in [3] the authors propose to mix the  $p$ -adic point-counting method with an early-abort strategy à la Schoof. Indeed for a small prime  $\ell$ , it is possible to decide quickly whether the group order is divisible by  $\ell$ , and if so to switch to another curve without running the  $p$ -adic algorithm. The size of the largest  $\ell$  for which we do this depends on the relative costs of the point-counting algorithm and the early-abort. Since [3], the cost of point-counting has been greatly reduced, and the number of  $\ell$  to consider must have diminished accordingly. Therefore it would be nice to also have new ideas in the early-abort stage to obtain another speed-up in the curve construction.

## Acknowledgments

We thank Takakazu Satoh, Robert Harley and François Morain for valuable comments and remarks.

## References

1. I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, volume 265 of *London Math. Soc. Lecture Note Ser.* Cambridge University Press, 1999.
2. M. Fouquet, P. Gaudry, and R. Harley. An extension of Satoh's algorithm and its implementation. *J. Ramanujan Math. Soc.*, 15:281–318, 2000.
3. M. Fouquet, P. Gaudry, and R. Harley. Finding secure curves with the Satoh-FGH algorithm and an early-abort strategy. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 14–29. Springer-Verlag, 2001.
4. T. Granlund. *The GNU Multiple Precision arithmetic library – 4.0.1*. Swox AB, 2002. distributed at <http://swox.com/gmp/>.
5. R. Harley. Counting points with the arithmetic-geometric mean (joint work with J.-F. Mestre and P. Gaudry). Eurocrypt 2001, Rump session.
6. A. Karp and P. Markstein. High precision division and square root. Technical Report 93-93-42, HP Labs, October 1994.
7. H. Kim, J. Park, J. Cheon, J. Park, J. Kim, and S. Hahn. Fast elliptic curve point counting using Gaussian normal basis. In C. Fieker and D. R. Kohel, editors, *ANTS-V*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 292–307. Springer-Verlag, 2002.
8. T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, 15:247–270, 2000.
9. T. Satoh. On  $p$ -adic point counting algorithms for elliptic curves over finite fields. In C. Fieker and D. R. Kohel, editors, *ANTS-V*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 43–66. Springer-Verlag, 2002.
10. T. Satoh, B. Skjernaa, and Y. Taguchi. Fast computation of canonical lifts of elliptic curves and its application to point counting. Preprint 2001.
11. R. Schoof. Elliptic curves over finite fields and the computation of square roots mod  $p$ . *Math. Comp.*, 44:483–494, 1985.
12. B. Skjernaa. Satoh's algorithm in characteristic 2. To appear in *Math. Comp.*
13. F. Vercauteren, B. Preneel, and J. Vandewalle. A memory efficient version of Satoh's algorithm. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 1–13. Springer-Verlag, 2001.
14. J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

# A General Formula of the $(t, n)$ -Threshold Visual Secret Sharing Scheme

Hiroki Koga

Faculty of Engineering Mechanics and Systems, University of Tsukuba, 1-1-1  
Tennoudai, Tsukuba-shi, Ibaraki 305-8573, Japan,  
`koga@esys.tsukuba.ac.jp`

**Abstract.** This paper provides a new method for construction of the generating (or basis) matrices of the  $(t, n)$ -threshold visual secret sharing scheme  $((t, n)$ -VSSS) for any  $n \geq 2$  and  $2 \leq t \leq n$ . We show that there exists a bijection between a set of generating matrices of the  $(t, n)$ -VSSS and a set of homogeneous polynomials of degree  $n$  satisfying a certain property. We also show that the set of homogeneous polynomials is identified with a set of lattice points in a linear space of dimension  $n - t + 1$  with explicitly expressed bases. These results yields a general formula of the generating matrices of the  $(t, n)$ -VSSS. The formula is not only theoretically of interest but also enables us to obtain efficient generating matrices that have been unknown.

## 1 Introduction

The visual secret sharing scheme (VSSS) is a new paradigm of the secret sharing proposed by Naor and Shamir [14]. Letting  $\mathcal{P} = \{1, 2, \dots, n\}$  be a set of participants, in the VSSS a black-white secret image is encrypted to  $n$  black-white images called shares. The VSSS has a property that, while a qualified set of participants can reproduce a secret image only by stacking all of their shares, a forbidden subset of participants can obtain no information on the secret image from their shares. If every  $S \subseteq \mathcal{P}$  with  $|S| \geq t$  is qualified and every  $S \subset \mathcal{P}$  with  $|S| \leq t - 1$  is forbidden for some  $2 \leq t \leq n$ , we call such a VSSS the  $(t, n)$ -VSSS, where  $|S|$  denotes the cardinality of  $S$ .

In this paper we focus on the  $(t, n)$ -VSSS. Literatures on the  $(t, n)$ -VSSS for black-white images can be classified into the following categories:

1. Construction of the optimal  $(n, n)$ -VSSS: [14].
2. Construction of the optimal  $(t, n)$ -VSSS in a certain class: [2] (for  $t = 2$ ), [3] (for  $t = 3, 4, 5, n - 1$ ).
3. Developing algorithms to find a non-optimal  $(t, n)$ -VSSS without optimality: [6], [9].
4. Giving examples of  $(t, n)$ -VSSS: [4], [5], [12], [13], [14].
5. Introducing another notion of optimality: [1], [7], [15].
6. Formulating the problem of finding the optimal  $(t, n)$ -VSSS as a linear programming problem: [3], [8], [11].

Here, the optimality of the  $(t, n)$ -VSSS is usually defined in terms of the clearness of the reproduced secret image obtained by stacking arbitrary  $t$  shares. In the literature above, however, the most important and fascinating problem of finding the optimal  $(t, n)$ -VSSS for arbitrary  $n \geq 2$  and  $2 \leq t \leq n$  still remains unsolved.

The  $(t, n)$ -VSSS is realized by using a pair of matrices  $(X_0, X_1)$  called generating matrices (though  $(X_0, X_1)$  is sometimes called basis matrices, we use a different terminology in order to avoid confusion). In this paper we propose a simple method for obtaining pairs of generating matrices of the  $(t, n)$ -VSSS that is valid for all  $n \geq 2$  and  $2 \leq t \leq n$ . The polynomial representation of generating matrices, which was first proposed by [10] and was extended by [12, 13], gives a key to the method. We show that a pair of generating matrices in a certain class can be identified with a lattice point in a linear space of homogeneous polynomials of dimension  $n - t + 1$ . More precisely, letting  $e_{t,n}^{(i)}$ ,  $i = 0, 1, \dots, n - t$ , be the bases of the linear space, for each  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}_{n-t+1}$ , where  $\mathcal{B}_{n-t+1}$  is the collection of all  $(\beta_0, \beta_1, \dots, \beta_{n-t})$  satisfying  $\beta_i \in \mathbf{Z}$  for all  $i = 0, 1, \dots, n - t$  and  $\sum_{i=0}^{n-t} \beta_i > 0$ , we can identify  $f = \sum_{i=0}^{n-t} e_{t,n}^{(i)} \beta_i$  as a pair of generating matrices. In addition, if we apply a simple operation to such  $f$ , we can obtain more efficient pair of generating matrices each of which belongs to a class of matrices that is often treated.

We can use the proposed method for obtaining suboptimal pairs of generating matrices. The optimality can be defined in arbitrary sense, that is, we can maximize the relative difference [14] or minimize the number of subpixels. We have only to consider a finite subset  $\mathcal{B}'_{n-t+1} \subset \mathcal{B}_{n-t+1}$  and exhaustively search for a pair of generating matrices in  $\mathcal{B}'_{n-t+1}$  that is the most desirable. We checked that this search is realistic if  $n \leq 9$  and found interesting examples of the  $(t, n)$ -VSSS that have been unknown.

This paper is organized as follows. In Section 2 we first define the  $(t, n)$ -VSSS mathematically. Then, we introduce important classes of matrices called column-permuting matrices (CPMs) [10] and different permuting matrices (DPMs) [12]. We explain several properties on concatenations of CPMs or DPMs. Section 3 is devoted to description of main results of this paper. We first show that there exists a bijection from the pairs of matrices realizing the  $(t, n)$ -VSSS to the set of homogeneous polynomials of degree  $n$  satisfying a certain property. We next show that for any  $n \geq 2$  and  $2 \leq t \leq n$  such homogeneous polynomials are regarded as lattice points of a linear space of dimension  $n - t + 1$ . These results mean that, surprisingly, any one of such lattice points yields a pair of generating matrices of the  $(t, n)$ -VSSS. We also give suboptimal pairs of generating matrices of the  $(t, n)$ -VSSS obtained for all  $n \leq 9$  that was found by computer search.

## 2 Visual Secret Sharing Scheme

### 2.1 Definition of the Visual Secret Sharing Scheme

Let  $\mathcal{P} = \{1, 2, \dots, n\}$  be a set of participants, where  $n \geq 2$ . Denote the set composed by all the subsets of  $\mathcal{P}$  by  $2^{\mathcal{P}}$ . Given an  $n \times m$  Boolean matrix  $X$

and an  $S \in 2^{\mathcal{P}}$ , we define  $X[S]$  as the  $|S| \times m$  matrix that is the restriction of  $X$  to the rows specified by  $S$ . The “or” of all the rows in  $X[S]$  is denoted by  $\text{OR}(X[S])$ . In addition, the Hamming weight of  $\text{OR}(X[S])$  is denoted by  $h(\text{OR}(X[S]))$ . Letting  $t$  be an arbitrary integer satisfying  $2 \leq t \leq n$ , we define the  $(t, n)$ -threshold visual secret sharing scheme  $((t, n)$ -VSSS for short) in the following way:

**Definition 1 (Naor and Shamir [14]).** Let  $\mathcal{C}_0$  and  $\mathcal{C}_1$  be collections of  $n \times m$  matrices. We say that a pair  $(\mathcal{C}_0, \mathcal{C}_1)$  forms the  $(t, n)$ -VSSS if  $(\mathcal{C}_0, \mathcal{C}_1)$  satisfies both of the following two conditions:

1. There exist constants  $d > 0$  and  $\alpha > 0$  satisfying:
  - (a) For any  $S \in 2^{\mathcal{P}}$  with  $|S| = t$ ,  $h(\text{OR}(X[S])) \leq d - \alpha m$  for all  $X \in \mathcal{C}_0$ .
  - (b) For any  $S \in 2^{\mathcal{P}}$  with  $|S| = t$ ,  $h(\text{OR}(X[S])) \geq d$  for all  $X \in \mathcal{C}_1$ .
2. For an  $S \in 2^{\mathcal{P}}$  and  $i = 0, 1$  define  $\mathcal{D}_i[S]$  as the collection of  $X[S]$ ,  $X \in \mathcal{C}_i$ . Then, for any  $S \in 2^{\mathcal{P}}$  with  $|S| < t$   $\mathcal{D}_0[S]$  and  $\mathcal{D}_1[S]$  are indistinguishable in the sense that they contain the same matrices with the same frequencies.

A secret image, which is assumed to be a black-white image, is encrypted into  $n$  images called *shares* in the following way. In fact, every pixel in a secret image is encrypted as  $m$  pixels called *subpixels* in each share. We first choose an element  $X \in \mathcal{C}_0$  ( $X \in \mathcal{C}_1$ ) randomly with uniform distribution if a pixel to be encrypted is white (black). Then, for  $i = 1, 2, \dots, n$  we encrypt the pixel as the  $m$  subpixels specified by the  $i$ -th row of  $X$ . This encryption is repeated until all the pixels in a secret image are encrypted. We assume that the  $i$ -th share is distributed to the participant  $i$  for  $i = 1, 2, \dots, n$ .

Condition 1-(a) in Definition 1 guarantees that for any  $S \in 2^{\mathcal{P}}$  with  $|S| = t$  a black-white secret image is reproduced only by stacking all of the shares specified by  $S$ . When we stack arbitrary  $t$  shares in an arbitrary order, we can perceive a gap of the Hamming weights more than  $\alpha m$  consisting in stacked  $m$  subpixels. That is, the  $m$  stacked subpixels corresponding to a white pixel in the secret image look brighter than the  $m$  stacked subpixels corresponding to a black pixel. Here, the parameter  $\alpha$  is called the *relative difference* [14]. In general, the greater  $\alpha$  becomes, the clearer we can perceive the secret image. On the other hand, condition 2 in Definition 1 means that no information on the secret image is revealed from the shares specified by  $S$  for any  $S \in 2^{\mathcal{P}}$  with  $|S| \leq t - 1$ . In fact, if  $|S| \leq t - 1$ , the participants in  $S$  can obtain no information on the color of a pixel because both  $\mathcal{D}_0[S]$  and  $\mathcal{D}_1[S]$  contain  $X[S]$  with the same frequencies.

It is often that  $\mathcal{C}_0$  and  $\mathcal{C}_1$  are constructed from all the permutations of rows of two matrices  $X_0$  and  $X_1$ . We call such matrices the *generating matrices*. Though such matrices are sometimes called the basis matrices rather than the generating matrices [23], we call  $(X_0, X_1)$  a pair of generating matrices in this paper because we use the term “basis” for expressing a different, but an ordinary, notion. Throughout this paper we consider construction of the  $(t, n)$ -VSSS using a pair of generating matrices. See [14] for examples of generating matrices.

## 2.2 Polynomial Representation of Generator Matrices

Hereafter, we consider the generating matrices that belong to a certain class. We define two classes of matrices called the column-permuting matrices (CPMs) [10] and the different permuting matrices (DPMs) [12].

Consider a Boolean vector  $V = [v_1, v_2, \dots, v_n]^T$  with  $n$  components, where the superscript  $T$  denotes the transpose. We can obtain  $n!$  vectors from all the permutations (permitting multiplicity) of components of  $V$ . The  $n \times n!$  matrix  $M_n(v_1, v_2, \dots, v_n)$  containing all of such  $n!$  vectors as rows is called a column-permuting matrix (CPM) of order  $n$  [10]. For the case of  $n = 3$  and  $V = [0, 0, 1]^T$ ,  $M_3(0, 0, 1)$  can be expressed as

$$M_3(0, 0, 1) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (1)$$

(In order to avoid confusion, readers may consider  $M_3(v_1, v_2, v_3)$  with distinct  $v_1, v_2$  and  $v_3$  and set  $v_1 = 0$ ,  $v_2 = 0$  and  $v_3 = 1$ .) We regard two CPMs as identical if an adequate permutation of rows of one equals to the other. We represent the CPM obtained from a vector with  $k$  1's and  $n - k$  0's as the monomial  $a^{n-k}z^k$ , where  $a$  and  $z$  are the symbols corresponding to 0 and 1, respectively. For example,  $M_3(0, 0, 1)$  in (1) is represented as  $a^2z$ .

Next, we consider concatenations of CPMs. Letting  $M_n(u_1, u_2, \dots, u_n)$  and  $M_n(v_1, v_2, \dots, v_n)$  be two CPMs with  $v_i, u_i \in \{0, 1\}$  for all  $i = 1, 2, \dots, n$ , we denote the concatenation of  $M_n(u_1, u_2, \dots, u_n)$  and  $M_n(v_1, v_2, \dots, v_n)$  by  $M_n(u_1, u_2, \dots, u_n) \odot M_n(v_1, v_2, \dots, v_n)$ . Here, we regard  $M_n(u_1, u_2, \dots, u_n) \odot M_n(v_1, v_2, \dots, v_n)$  as the  $n \times (2n!)$  matrix containing all the permutations (permitting multiplicity) of two Boolean vectors  $[u_1, u_2, \dots, u_n]^T$  and  $[v_1, v_2, \dots, v_n]^T$ . We regard two concatenations of CPMs as identical if an adequate permutation of rows of one equals the other. Letting  $[u_1, u_2, \dots, u_n]^T$  be a Boolean vector with  $k$  1's and  $n - k$  0's and  $[v_1, v_2, \dots, v_n]^T$  a Boolean vector with  $l$  1's and  $n - l$  0's, we represent  $M_n(u_1, u_2, \dots, u_n) \odot M_n(v_1, v_2, \dots, v_n)$  as the polynomial  $a^{n-k}z^k + a^{n-l}z^l$ . That is, the concatenation of matrices is represented by using  $+$  in the polynomial representation. In particular, for the case of  $k = l$  we express  $a^{n-k}z^k + a^{n-k}z^k$  as  $2a^{n-k}z^k$  for short. Obviously, any concatenation of CPMs of order  $n$  is represented as a homogeneous polynomial of  $a$  and  $z$  of degree  $n$ . In addition, it is important to notice that two concatenations of CPMs are identical if and only if the polynomial representation of one is equal to the other in the ordinary sense. For example, a concatenation of CPMs  $M_3(0, 0, 0) \odot M_3(0, 1, 1) \odot M_3(0, 1, 1) \odot M_3(1, 1, 1)$ , which is represented as  $a^3 + 2az^2 + z^3$ , is identical with another concatenation of CPMs  $M_3(0, 1, 1) \odot M_3(1, 1, 1) \odot M_3(0, 1, 1) \odot M_3(0, 0, 0)$  that also has the polynomial representation  $az^2 + z^3 + az^2 + a^3 = a^3 + 2az^2 + z^3$ .

It is important to notice that we can represent the operation in which we eliminate an arbitrary row from a CPM  $M_n(v_1, v_2, \dots, v_n)$  as application of the partial differential operator  $\psi \stackrel{\text{def}}{=} \frac{\partial}{\partial a} + \frac{\partial}{\partial z}$  to the polynomial representation of

$M_n(v_1, v_2, \dots, v_n)$ . For example, if we eliminate the the third row of  $M_3(0, 0, 1)$  in (1), we have  $M_2(0, 0) \odot M_2(0, 1) \odot M_2(0, 1)$ . This operation is represented as  $\psi(a^2z) = a^2 + 2az$  in the polynomial representation. The definition of the CPM guarantees that we can obtain the same matrix if we eliminate either the first row or the second row instead of the third row. In the same way, the operation eliminating  $j$  arbitrary rows from a CPM is represented as application of  $\psi$  to its polynomial representation repeatedly for  $j$  times. The repeated application of  $\psi$  for  $j$  times is denoted by  $\psi^j$ . It is obvious that the same property on the elimination of rows also holds for concatenations of CPMs.

Next, we define the different permuting matrix (DPM). While [3, 6, 9] use different terminology for the same class of matrices, we follow the terminology given in [12]. Consider a Boolean vector  $V = [v_1, v_2, \dots, v_n]^T$ . Suppose that  $V$  contains  $k$  1's and  $n - k$  0's as its components. Then, we can obtain  $\binom{n}{k}$  different vectors from all the permutations of components of  $V$ . The  $n \times \binom{n}{k}$  matrix containing all of such  $\binom{n}{k}$  vectors as rows is called a different permuting matrix (DPM) of order  $n$  and is denoted by  $N_n(v_1, v_2, \dots, v_n)$ . For the case of  $n = 3$ ,  $N_3(0, 0, 0)$  and  $N_3(0, 0, 1)$  are written as

$$N_3(0, 0, 0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad N_3(0, 0, 1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2)$$

It is important to notice that  $M_3(0, 0, 1)$  in (1) satisfies  $M_3(0, 0, 1) = N_3(0, 0, 1) \odot N_3(0, 0, 1)$  (recall that rows of  $M_3(0, 0, 1)$  can be permuted adequately). More generally, for  $[v_1, v_2, \dots, v_n]^T$  containing  $k$  1's and  $n - k$  0's, it is easy to verify that  $M_n(v_1, v_2, \dots, v_n)$  is the concatenation of  $(n - k)!k!$   $N_n(v_1, v_2, \dots, v_n)$ 's. This motivates us to represent  $N_n(v_1, v_2, \dots, v_n)$  as the monomial  $\frac{a^{n-k}z^k}{(n-k)!k!}$  [12].

In particular, for the cases of  $k = 0$  and  $k = n$  we use the representations  $\frac{a^n}{n!}$  and  $\frac{z^n}{n!}$ , respectively. We also use  $+$  for denoting concatenation of DPMs. Then, it obviously follows that eliminating an arbitrary row from a DPM is represented as application of  $\psi = \frac{\partial}{\partial a} + \frac{\partial}{\partial z}$  to the monomial representation of the DPM. In fact, if we eliminate the third row from  $N_3(0, 0, 1)$ , which is represented as  $\frac{a^2z}{2!1!}$ , we have the concatenation of DPMs represented as  $\psi(\frac{a^2z}{2!1!}) = az + \frac{a^2}{2!}$ . In addition, eliminating  $j$  arbitrary rows from a DPM is represented as application of  $\psi^j$  to its monomial representation. It is obvious that the same property on the elimination of rows holds for concatenations of DPMs.

Now, we introduce the following four sets of homogeneous polynomials:

$$\mathcal{H}_n = \left\{ \sum_{i=0}^n \gamma_i a^{n-i} z^i : \gamma_i \in \mathbf{Z} \text{ for all } i = 0, 1, \dots, n \right\}, \quad (3)$$

$$\mathcal{H}_n^+ = \left\{ \sum_{i=0}^n \gamma_i a^{n-i} z^i : \gamma_i \in \mathbf{Z} \text{ and } \gamma_i > 0 \text{ for all } i = 0, 1, \dots, n \right\}, \quad (4)$$

$$\mathcal{K}_n = \left\{ \sum_{i=0}^n \gamma_i \frac{a^{n-i} z^i}{(n-i)!i!} : \gamma_i \in \mathbf{Z} \text{ for all } i = 0, 1, \dots, n \right\}, \quad (5)$$



$$\mathcal{K}_n^+ = \left\{ \sum_{i=0}^n \gamma_i \frac{a^{n-i} z^i}{(n-i)! i!} : \gamma_i \in \mathbf{Z} \text{ and } \gamma_i > 0 \text{ for all } i = 0, 1, \dots, n \right\}, \quad (6)$$

where  $\mathbf{Z}$  denotes the set of all integers. Then, as summary, we have the following proposition.

- Proposition 1.** (a) Any concatenation of CPMs (DPMs) of order  $n$  is expressed as an element in  $\mathcal{H}_n^+$  ( $\mathcal{K}_n^+$ ). Conversely, any element in  $\mathcal{H}_n^+$  ( $\mathcal{K}_n^+$ ) is interpreted as a concatenation of CPMs (DPMs) of order  $n$ .
- (b) Let  $X$  be any concatenation of CPMs (DPMs) with the polynomial representation  $f \in \mathcal{H}_n^+$  ( $f \in \mathcal{K}_n^+$ ). Then, for any  $1 \leq j \leq n-1$  the polynomial representation of the matrix obtained by eliminating arbitrary  $j$  rows from  $X$  is given by  $\psi^j f$ .

Then, we have the following theorem. While the primary version of Theorem 1(a) was given by Koga, Iwamoto and Yamamoto [10] for the  $(t, n)$ -VSSS of color images, Kuwakado and Tanaka [12] pointed out that Theorem 1(b) holds for the case of  $(t, n)$ -VSSS of black-white images. Proof of Theorem 1 is given in Appendix A for readers' convenience.

**Theorem 1.** (a) Suppose that  $f_0 \in \mathcal{H}_n^+$  and  $f_1 \in \mathcal{H}_n^+$  satisfy

$$\psi^{n-t+1} f_0 = \psi^{n-t+1} f_1 \quad (7)$$

and

$$\psi^{n-t} f_0|_{z=0} = C_0 a^t, \quad \psi^{n-t} f_1|_{z=0} = C_1 a^t \quad (8)$$

for some nonnegative integers  $C_0$  and  $C_1$  with  $C_0 > C_1$ . Define  $X_0$  and  $X_1$  as the concatenations of CPMs with the polynomial expressions  $f_0$  and  $f_1$ , respectively. Then,  $(X_0, X_1)$  becomes a pair of generating matrices of the  $(t, n)$ -VSSS.

(b) Suppose that  $g_0 \in \mathcal{K}_n^+$  and  $g_1 \in \mathcal{K}_n^+$  satisfy

$$\psi^{n-t+1} g_0 = \psi^{n-t+1} g_1 \quad (9)$$

and

$$\psi^{n-t} g_0|_{z=0} = C_0 \frac{a^t}{t!}, \quad \psi^{n-t} g_1|_{z=0} = C_1 \frac{a^t}{t!} \quad (10)$$

for some nonnegative integers  $C_0$  and  $C_1$  with  $C_0 > C_1$ . Define  $X_0$  and  $X_1$  as the concatenations of DPMs with the polynomial expressions  $g_0$  and  $g_1$ , respectively. Then,  $(X_0, X_1)$  becomes a pair of generating matrices of the  $(t, n)$ -VSSS.

We conclude this section with introducing two more notions. First, we define the decomposition of an element in  $\mathcal{H}_n$  or  $\mathcal{K}_n$ . If an  $f \in \mathcal{H}_n$  is written as  $f = f^+ - f^-$ , where  $f^+$  and  $f^-$  belong to  $\mathcal{H}_n^+ \cup \{0\}$  and  $f^+$  and  $f^-$  contain no term in common, we call  $f = f^+ - f^-$  the decomposition of  $f$ . For example, if  $f = a^2 z - a z^2 + z^3 \in \mathcal{H}_3$ , we have  $f^+ = a^2 z + z^3$  and  $f^- = a z^2$ . Note that the decomposition is unique and  $f^+$  ( $f^-$ ) equals zero if all the terms in  $f$  have

negative (positive) coefficients. The decomposition of  $g \in \mathcal{K}_n$  is defined in the same way. That is, if  $g = \frac{a^3}{3!} - \frac{a^2 z}{2!1!} + 2 \frac{z^3}{3!} \in \mathcal{K}_3$ , we have  $g^+ = \frac{a^3}{3!} + 2 \frac{z^3}{3!}$  and  $g^- = \frac{a^2 z}{2!1!}$ . Next, we define the *norm* of  $f \in \mathcal{H}_n$ . Suppose that  $f$  is expressed as  $f = \sum_{i=0}^n \gamma_i a^{n-i} z^i$ . Then, the norm  $\|f\|$  of  $f$  is defined by  $\|f\| = \sum_{i=0}^n |\gamma_i|$ , where  $|\gamma_i|$  denotes the absolute value of  $\gamma_i$ . Clearly,  $\|f\| \geq 0$  for all  $f \in \mathcal{H}_n$  and  $\|f\| = 0$  if and only if  $f = 0$ . It is clear that for the matrix  $X$  with a polynomial expression  $f \in \mathcal{H}_n^+$ ,  $\|f\|$  means the number of CPMs contained in  $X$ .

### 3 Main Results

#### 3.1 Characterization of the $(t, n)$ -VSSS as a Vector Space

Theorem 1(a) guarantees that, if we can find  $f_0 \in \mathcal{H}_n^+$  and  $f_1 \in \mathcal{H}_n^+$  satisfying (7) and (8), we obtain a pair of generating matrices  $(X_0, X_1)$  of the  $(t, n)$ -VSSS, where  $X_0$  and  $X_1$  are the concatenations of CPMs corresponding to  $f_0$  and  $f_1$ , respectively. However, Theorem 1 does not tell us at all how we can find such  $f_0$  and  $f_1$ . Since  $\psi^{n-t+1}$  and  $\psi^{n-t}$  are linear, the homogeneous polynomial  $f \in \mathcal{H}_n$  defined by  $f = f_0 - f_1$  satisfies  $\psi^{n-t+1} f = 0$  and  $\psi^{n-t} f|_{z=0} = C a^t$  for some integer  $C > 0$ . This motivates us to define the following subsets of  $\mathcal{H}_n$  and  $\mathcal{K}_n$ :

$$\begin{aligned}\mathcal{F}_{t,n} &= \{f \in \mathcal{H}_n : \psi^{n-t+1} f = 0 \text{ and } \psi^{n-t} f|_{z=0} = C a^t \text{ for some integer } C > 0\}, \\ \mathcal{G}_{t,n} &= \{g \in \mathcal{K}_n : \psi^{n-t+1} g = 0 \text{ and } \psi^{n-t} g|_{z=0} = C \frac{a^t}{t!} \text{ for some integer } C > 0\}.\end{aligned}$$

We also define the sets of pairs of matrices by

$$\begin{aligned}\mathcal{M}_{t,n} &= \{(X_0, X_1) : X_0 \text{ and } X_1 \text{ satisfy all of } (A_1), (B_1), (C_1), (D_1)\}, \\ \mathcal{N}_{t,n} &= \{(X_0, X_1) : X_0 \text{ and } X_1 \text{ satisfy all of } (A_2), (B_2), (C_1), (D_1)\},\end{aligned}$$

where conditions  $(A_1)$ ,  $(A_2)$ ,  $(B_1)$ ,  $(B_2)$ ,  $(C_1)$  and  $(D_1)$  are given as follows:

- (A<sub>1</sub>) both  $X_0$  and  $X_1$  are concatenations of CPMs,
- (A<sub>2</sub>) both  $X_0$  and  $X_1$  are concatenations of DPMs,
- (B<sub>1</sub>)  $X_0$  and  $X_1$  contain no CPM in common,
- (B<sub>2</sub>)  $X_0$  and  $X_1$  contain no DPM in common,
- (C<sub>1</sub>)  $X_0[S] = X_1[S]$  for any  $S \in 2^P$  with  $|S| = t - 1$ , where the equality  $X_0[S] = X_1[S]$  is interpreted in the sense that  $X_1[S]$  coincides  $X_0[S]$  by an adequate permutation of rows,
- (D<sub>1</sub>)  $h(\text{OR}(X_0[S])) < h(\text{OR}(X_1[S]))$  for any  $S \in 2^P$  with  $|S| = t$ , where  $h(\cdot)$  denotes the Hamming weight.

That is,  $\mathcal{M}_{t,n}$  ( $\mathcal{N}_{t,n}$ ) is the set of all the pairs of generating matrices obtained by concatenations of CPMs (DPMs) containing no CPM (DPM) in common. Then, we have the following theorem that is a stronger version of Theorem 1.

**Theorem 2.** *For any  $n \geq 2$  and  $2 \leq t \leq n$ , there exist bijections  $\varphi : \mathcal{M}_{t,n} \rightarrow \mathcal{F}_{t,n}$  and  $\sigma : \mathcal{N}_{t,n} \rightarrow \mathcal{G}_{t,n}$ .*

*Proof.* We only prove the existence of a bijection  $\varphi : \mathcal{M}_{t,n} \rightarrow \mathcal{F}_{t,n}$  below because the existence of  $\sigma : \mathcal{N}_{t,n} \rightarrow \mathcal{G}_{t,n}$  can be proved in the same way.

Suppose that  $(X_0, X_1) \in \mathcal{M}_{t,n}$ . Since  $X_0$  and  $X_1$  are assumed to be concatenations of CPMs, Proposition 1 guarantees that there exist unique  $f_0 \in \mathcal{H}_n^+$  and  $f_1 \in \mathcal{H}_n^+$  corresponding to  $X_0$  and  $X_1$ , respectively. In addition, we note that the converse of Theorem 1 is also true. That is, such  $f_0$  and  $f_1$  satisfy (7) and (8). We define  $f$  by  $f = f_0 - f_1$ . Clearly,  $f$  belongs to  $\mathcal{H}_n$  and satisfies

$$\psi^{n-t+1}f = 0 \quad \text{and} \quad \psi^{n-t}f|_{z=0} = (C_1 - C_2)a^t \quad (11)$$

due to the linearity of  $\psi^{n-t+1}$  and  $\psi^{n-t}$ . Since  $C_1 > C_2$  from Definition 1 and the definitions of  $f_0, f_1$  and  $f$ , (11) guarantees that  $f \in \mathcal{F}_{t,n}$ . We define  $\varphi$  as the mapping that maps a pair  $(X_0, X_1) \in \mathcal{M}_{t,n}$  to  $f = f_0 - f_1 \in \mathcal{F}_{t,n}$ , where  $f_0$  and  $f_1$  are the polynomial representations of  $X_0$  and  $X_1$ , respectively.

First, we prove that  $\varphi$  is one-to-one. Assume that  $(X_0, X_1) \in \mathcal{M}_{t,n}$  and  $(\tilde{X}_0, \tilde{X}_1) \in \mathcal{M}_{t,n}$  satisfy  $\varphi(X_0, X_1) = \varphi(\tilde{X}_0, \tilde{X}_1)$ . Let  $f_0, f_1, \tilde{f}_0$  and  $\tilde{f}_1$  be the polynomial expressions of  $X_0, X_1, \tilde{X}_0$  and  $\tilde{X}_1$ , respectively. Note that, since  $(X_0, X_1) \in \mathcal{M}_{t,n}$ ,  $f_0$  and  $f_1$  contain no term in common due to the definition of  $\mathcal{M}_{t,n}$ . Similarly,  $\tilde{f}_0$  and  $\tilde{f}_1$  contain no term in common as well. It is important to notice that  $\varphi(X_0, X_1) = \varphi(\tilde{X}_0, \tilde{X}_1)$  means that  $f_0 - f_1 = \tilde{f}_0 - \tilde{f}_1$ , i.e.,

$$f_0 - \tilde{f}_0 = f_1 - \tilde{f}_1. \quad (12)$$

Now, define  $h$  by  $h = f_0 - \tilde{f}_0$ . Clearly,  $h \in \mathcal{H}_n$  because both  $f_0$  and  $\tilde{f}_0$  belong to  $\mathcal{H}_n$ . Denoting the decomposition of  $h$  by  $h = h^+ - h^-$ , (12) leads to

$$\begin{cases} f_0 + h^- = \tilde{f}_0 + h^+, \\ f_1 + h^- = \tilde{f}_1 + h^+. \end{cases} \quad (13)$$

Since  $h^+$  and  $h^-$  contain no term in common due to the definition of the decomposition, (13) means that both  $f_0$  and  $f_1$  contain  $h^+$  in common. This implies that  $h^+ = 0$  because  $f_0$  and  $f_1$  contain no term in common by assumption. Similarly, we obtain  $h^- = 0$ , and therefore, we have  $h = 0$ . By combining  $h = 0$  with (12), we have  $f_0 - f_1 = \tilde{f}_0 - \tilde{f}_1$ , i.e.,  $(X_0, X_1) = (\tilde{X}_0, \tilde{X}_1)$ , which shows that  $\varphi$  is one-to-one.

Next, we prove that  $\varphi$  is onto. To this end, fix an  $f \in \mathcal{F}_{t,n}$  arbitrarily. Then, it holds that  $\psi^{n-t+1}f = 0$  and  $\psi^{n-t}f|_{z=0} = Ca^t$  for some integer  $C > 0$ . Letting  $f = f^+ - f^-$  be the decomposition of  $f$ , it follows that  $\psi^{n-t+1}f^+ = \psi^{n-t+1}f^-$  and

$$\psi^{n-t}f|_{z=0} = \psi^{n-t}f^+|_{z=0} - \psi^{n-t}f^-|_{z=0} = Ca^t \quad (14)$$

owing to the linearity of  $\psi^{n-t+1}$  and  $\psi^{n-t}$ . In addition, since  $f^+$  and  $f^-$  belong to  $\mathcal{H}_n^+ \cup \{0\}$ , there exist integers  $C_0 \geq 0$  and  $C_1 \geq 0$  such that  $\psi^{n-t}f^+|_{z=0} = C_0a^t$  and  $\psi^{n-t}f^-|_{z=0} = C_1a^t$ . In view of (14),  $C_0$  and  $C_1$  satisfy  $C_0 = C_1 + C > C_1$ . Therefore, by virtue of Theorem 1(a), the pair of matrices  $(X_0, X_1)$  corresponding to  $(f^+, f^-)$  satisfies  $(X_0, X_1) \in \mathcal{M}_{t,n}$ .  $\square$

Since Theorem 2 guarantees the existence of a bijection  $\varphi : \mathcal{M}_{t,n} \rightarrow \mathcal{F}_{t,n}$ , we can know more about  $\mathcal{M}_{t,n}$  by developing properties of  $\mathcal{F}_{t,n}$ . The following lemma characterizes a key property of a set including  $\mathcal{F}_{t,n}$ .

**Lemma 1.** *Define*

$$\mathcal{E}_{t,n} = \{f \in \mathcal{R}_n : \psi^{n-t+1}f = 0\}, \quad (15)$$

where

$$\mathcal{R}_n = \left\{ \sum_{i=0}^n \gamma_i a^{n-i} z^i : \gamma_i \in \mathbf{R} \right\}$$

and  $\mathbf{R}$  denotes the set of all real numbers. Then,  $\mathcal{E}_{t,n}$  is a linear space of dimension  $n - t + 1$  with bases

$$e_{t,n}^{(i)} = a^{n-t-i} z^i (a - z)^t, \quad i = 0, 1, \dots, n - t. \quad (16)$$

*Proof.* Clearly, the linearity of  $\psi^{n-t+1}$  implies that  $\mathcal{E}_{t,n}$  is a linear space. We prove both  $\dim \mathcal{E}_{t,n} \geq n - t + 1$  and  $\dim \mathcal{E}_{t,n} \leq n - t + 1$ , where  $\dim \mathcal{E}_{t,n}$  denotes the dimension of  $\mathcal{E}_{t,n}$ . We can see that  $e_{t,n}^{(i)}$ ,  $0 \leq i \leq n - t$ , form bases of  $\mathcal{E}_{t,n}$  from the proof below.

First, we prove  $\dim \mathcal{E}_{t,n} \geq n - t + 1$ . We use the formula similar to the Leibniz formula

$$\psi^k(fg) = \sum_{j=0}^k \binom{k}{j} (\psi^{k-j}f)(\psi^jg) \quad (17)$$

for all  $k \geq 1$  and infinitely differentiable  $f$  and  $g$ , which can be easily proved by induction on  $k$ . Letting  $i$  an arbitrary integer with  $0 \leq i \leq n - t$ , it follows from (17) that

$$\begin{aligned} \psi^{n-t+1} e_{t,n}^{(i)} &= \psi^{n-t+1} (a^{n-t-i} z^i (a - z)^t) \\ &= \sum_{j=0}^{n-t+1} \binom{n-t+1}{j} (\psi^{n-t+1-j} (a^{n-t-i} z^i)) (\psi^j (a - z)^t). \end{aligned} \quad (18)$$

By noticing that  $\psi^j(a - z)^t = 0$  for all  $j \geq 1$ , (18) leads to

$$\psi^{n-t+1} e_{t,n}^{(i)} = \left[ \psi^{n-t+1} (a^{n-t-i} z^i) \right] (a - z)^t = 0 \quad (19)$$

where the last equality in (19) follows because  $\psi^{n-t+1}f = 0$  for any  $f \in \mathcal{R}_k$  with  $k < n - t + 1$ . Hence, we have  $\psi^{n-t+1} e_{t,n}^{(i)} = 0$  for all  $i = 0, 1, \dots, n - t$ .

We can verify that  $e_{t,n}^{(i)}$ ,  $0 \leq i \leq n - t$ , are linearly independent in the following way. Assume that there exist real numbers  $\beta_0, \beta_1, \dots, \beta_{n-t}$  satisfying

$$\sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)} = 0. \quad (20)$$

We notice that the greatest degree with respect to  $a$  on the left-hand side of (20) is at most  $n$  and a term including  $a^n$  appears only in  $e_{t,n}^{(0)}$ . This means that  $\beta_0 = 0$ . By repeating this argument, we have  $\beta_0 = \beta_1 = \cdots = \beta_{n-t} = 0$ . Hence, it turns out that  $e_{t,n}^{(i)}$ ,  $0 \leq i \leq n-t$ , are linearly independent. Consequently, we have established that  $\dim \mathcal{E}_{t,n} \geq n-t+1$ .

Next, we prove  $\dim \mathcal{E}_{t,n} \leq n-t+1$ . We prove that any  $f \in \mathcal{E}_{t,n}$  can be expressed as a linear combination of  $e_{t,n}^{(i)}$ ,  $0 \leq i \leq n-t$ . To this end, fix

$$f = \sum_{i=0}^n \gamma_i a^{n-i} z^i \in \mathcal{E}_{t,n} \quad (21)$$

arbitrarily, where  $\gamma_i \in \mathbf{R}$  for all  $i = 0, 1, \dots, n-t$ . Then, since among the bases  $e_{t,n}^{(i)}$ ,  $0 \leq i \leq n-t$ , the term including  $a^n$  is contained only in  $e_{t,n}^{(0)}$  and the coefficient of such a term in  $e_{t,n}^{(0)}$  is equal to 1,  $f$  can be written as

$$f = \gamma_0 e_{t,n}^{(0)} + \left( \sum_{i=1}^n \gamma_i a^{n-i} z^i - \gamma_0 e_{t,n}^{(0)} \right). \quad (22)$$

Notice that the greatest degree with respect to  $a$  of the second term in (22) is at most  $n-1$ . That is, we can rewrite (22) in the following form:

$$f = \gamma_0 e_{t,n}^{(0)} + \sum_{i=1}^n \gamma'_i a^{n-i} z^i, \quad (23)$$

where  $\gamma_i$ ,  $1 \leq i \leq n$ , are constants determined from  $\gamma_i$ ,  $1 \leq i \leq n-t$ , and  $e_{t,n}^{(0)}$ . By repeating this argument, we next have

$$f = \gamma_0 e_{t,n}^{(0)} + \gamma'_1 e_{t,n}^{(1)} + \sum_{i=2}^n \gamma''_i a^{n-i} z^i, \quad (24)$$

and finally have

$$f = \sum_{i=0}^{n-t} \tilde{\gamma}_i e_{t,n}^{(i)} + g, \quad (25)$$

where  $\tilde{\gamma}_i$ ,  $0 \leq i \leq n-t$ , are constants,  $g = z^{n-t+1}h$  and  $h \in \mathcal{R}_{t-1}$ . Here, we use the following lemma that is proved in Appendix B.

**Lemma 2.** *Let  $l$  be an arbitrary integer satisfying  $0 \leq l \leq n$ . If  $g \in \mathcal{R}_n$  can be written as  $g = z^l h$  for some  $h \in \mathcal{R}_{n-l}$  and satisfies  $\psi^l g = 0$ , then  $g = 0$ .*

Since it holds that  $\psi^{n-t+1} f = 0$  and  $\psi^{n-t+1} e_{t,n}^{(i)} = 0$  for all  $i = 0, 1, \dots, n-t$ , (25) implies that  $\psi^{n-t+1} g = 0$ . By applying Lemma 2 to  $g$  in (25), we have  $g = 0$ . This completes the proof of  $\dim \mathcal{E}_{t,n} \leq n-t+1$ .  $\square$

Now, we are ready to give the following theorem that characterizes  $\mathcal{F}_{t,n}$  as lattice points.

**Theorem 3.** For any  $n \geq 2$  and  $2 \leq t \leq n$ , it holds that

$$\mathcal{F}_{t,n} = \left\{ \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)} : \beta_i \in \mathbf{Z} \text{ for all } i = 0, 1, \dots, n-t \text{ and } \sum_{i=0}^{n-t} \beta_i > 0 \right\}. \quad (26)$$

*Proof.* We use the fact that  $e_{t,n}^{(i)}$ ,  $0 \leq i \leq n-t$ , satisfy  $\psi^{n-t} e_{t,n}^{(i)} = (n-t)!(a-z)^t$  and therefore

$$\psi^{n-t} e_{t,n}^{(i)}|_{z=0} = (n-t)! a^t \quad \text{for all } i = 0, 1, \dots, n-t, \quad (27)$$

which can be easily verified similarly to the method that develops  $\psi^{n-t+1} e_{t,n}^{(i)} = 0$  in (18) and (19). Let  $\mathcal{L}_{t,n}$  denote the set on the right-hand side of (26). We prove Theorem 3 by developing both  $\mathcal{F}_{t,n} \subseteq \mathcal{L}_{t,n}$  and  $\mathcal{L}_{t,n} \subseteq \mathcal{F}_{t,n}$ . Since  $\mathcal{F}_{t,n} \subset \mathcal{E}_{t,n}$ , an arbitrary  $f \in \mathcal{F}_{t,n}$  can be expressed as

$$f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)} + g \quad (28)$$

by using the same method that yields (25), where  $g = z^{n-t+1}h$  and  $h \in \mathcal{R}_{t-1}$ . If we apply Lemma 2 to  $g$  in (28), we have  $g = 0$ . In addition, it is important to notice that  $\beta_i \in \mathbf{Z}$  for all  $i = 0, 1, \dots, n-t$  because no division is included in the method. By applying  $\psi^{n-t}$  to both sides of (28) and set  $z = 0$ , we have

$$\psi^{n-t} f|_{z=0} = (n-t)! \left( \sum_{i=0}^{n-t} \beta_i \right) a^t \quad (29)$$

from (27). Since  $f \in \mathcal{F}_{t,n}$  satisfies  $\psi^{n-t} f|_{z=0} = C a^t$  for some integer  $C > 0$ , (29) implies that  $\sum_{i=0}^{n-t} \beta_i > 0$ . This establishes  $\mathcal{F}_{t,n} \subseteq \mathcal{L}_{t,n}$ .

Proof of  $\mathcal{L}_{t,n} \subseteq \mathcal{F}_{t,n}$  is easy. Fix an  $f \in \mathcal{L}_{t,n}$  arbitrarily. Since  $f \in \mathcal{L}_{t,n}$ ,  $f$  is expressed as  $f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)}$ , where  $\beta_i \in \mathbf{Z}$  for all  $i = 0, 1, \dots, n-t$  and  $\sum_{i=0}^{n-t} \beta_i > 0$ . Then, it immediately follows from Lemma 1, (27) and the linearity of  $\psi^{n-t+1}$  and  $\psi^{n-t}$  that

$$\psi^{n-t+1} f = \sum_{i=0}^{n-t} \beta_i (\psi^{n-t+1} e_{t,n}^{(i)}) = \sum_{i=0}^{n-t} \beta_i \cdot 0 = 0, \quad (30)$$

$$\psi^{n-t} f|_{z=0} = \sum_{i=0}^{n-t} \beta_i (\psi^{n-t} e_{t,n}^{(i)})|_{z=0} = (n-t)! \left( \sum_{i=0}^{n-t} \beta_i \right) a^t, \quad (31)$$

which shows that  $f \in \mathcal{F}_{t,n}$  because  $\sum_{i=0}^{n-t} \beta_i > 0$  by assumption.  $\square$

Now, define

$$\mathcal{B}_k = \left\{ (\beta_0, \beta_1, \dots, \beta_{k-1}) \in \mathbf{Z}^k : \sum_{i=0}^{k-1} \beta_i > 0 \right\}. \quad (32)$$

for  $k \geq 1$ . Then, Theorem 3 tells us that each  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}_{n-t+1}$  gives an element of  $f \in \mathcal{F}_{t,n}$ . Since Theorem 2 guarantees that there exists a bijection  $\varphi : \mathcal{M}_{t,n} \rightarrow \mathcal{F}_{t,n}$ , such that  $f \in \mathcal{F}_{t,n}$  yields a pair of generating matrices  $\varphi^{-1}(f) = (X_0, X_1) \in \mathcal{M}_{t,n}$ . The following corollary describes properties of such a pair of generating matrices.

**Corollary 1.** *Let  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}_{n-t+1}$  be arbitrarily given. Let  $(X_0, X_1) \in \mathcal{M}_{t,n}$  be the pair of generating matrices corresponding to  $f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)} \in \mathcal{F}_{t,n}$ . Then, the relative difference  $\alpha$  of  $(X_0, X_1)$  is given by*

$$\alpha = 2 \sum_{i=0}^{n-t} \beta_i \Big/ \left[ \binom{n}{t} \|f\| \right], \quad (33)$$

where  $\|f\|$  denotes the norm of  $f$ . In addition, the number of rows in  $X_0$  (or  $X_1$ ) is equal to  $\|f\| \cdot n!/2$ .

*Proof.* Recall that  $\psi^{n-t} f|_{z=0}$  is given by (31) for any  $f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)} \in \mathcal{F}_{t,n}$ . Equation (31) means that the relative difference of the pair of generating matrices  $\varphi^{-1}(f) = (X_0, X_1)$  is caused by  $(n-t)! \left( \sum_{i=0}^{n-t} \beta_i \right)$  CPMs each of which is represented as  $a^t$ . Since the CPM represented as  $a^t$  contains  $t!$  0's, the number of subpixels yielding relative difference is equal to  $W = (n-t)!t! \sum_{i=0}^{n-t} \beta_i$ .

Next, we evaluate the number of rows contained in  $X_0$  or  $X_1$ . Recall that, letting  $f = f^+ - f^-$  be the decomposition of  $f$ ,  $X_0$  and  $X_1$  have the polynomial representations  $f^+$  and  $f^-$ , respectively. Clearly, we have  $\|f\| = \|f^+\| + \|f^-\|$ . In addition, since  $f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)}$  and  $e_{t,n}^{(i)}|_{a=z=1} = 0$  for all  $i = 0, 1, \dots, n-t$ , setting  $a = z = 1$  in  $f = f^+ - f^-$  leads to  $\|f^+\| = \|f^-\|$ . Hence, it holds that  $\|f^+\| = \|f^-\| = \|f\|/2$ . Since both  $X_0$  and  $X_1$  are concatenations of  $\|f^+\| = \|f^-\|$  CPMs each of which has  $n!$  rows, the number of rows  $M$  of  $X_0$  and  $X_1$  turns out to satisfy  $M = \|f\| \cdot n!/2$ . Then, the claim of the corollary is immediate because  $\alpha = W/M$ .  $\square$

We have developed a method which enables us to construct a pair of generating matrix  $(X_0, X_1) \in \mathcal{M}_{t,n}$  from an  $f \in \mathcal{F}_{t,n}$ . In fact, letting  $f$  be an arbitrary element of  $\mathcal{F}_{t,n}$  and  $f = f^+ - f^-$  the decomposition of  $f$ ,  $X_0$  and  $X_1$  are concatenations of CPMs with the polynomial representations  $f^+$  and  $f^-$ , respectively. However, Corollary 1 tells us that the number of rows of such  $X_0$  and  $X_1$  can be large because they have  $\|f\| \cdot n!/2$  rows.

However, we can also develop a method for finding a pair of generating matrices with less number of rows. We make use of the fact that any CPM can be written as a concatenation of DPMs. To this end, we define

$$\mathcal{G}_{t,n}^* = \left\{ \sum_{i=0}^n \gamma_i \frac{a^{n-i} z^i}{(n-i)! i!} \in \mathcal{G}_{t,n} : \gcd\{\gamma_i : i = 0, 1, \dots, n-t\} = 1 \right\}, \quad (34)$$

where  $\gcd\{\gamma_i : i = 0, 1, \dots, n-t\}$  denotes the greatest common divisor  $\geq 1$ . In order to reduce the number of rows, we use the mapping  $\pi : \mathcal{F}_{t,n} \rightarrow \mathcal{G}_{t,n}^*$  given in the following proposition.

**Proposition 2.** For any  $n \geq 2$  and  $2 \leq t \leq n$  there exists a surjection  $\pi : \mathcal{F}_{t,n} \rightarrow \mathcal{G}_{t,n}^*$ .

*Proof.* We define  $\pi$  in the following way. Let  $f = \sum_{i=0}^n \gamma_i a^{n-i} z^i$  be an arbitrary element in  $\mathcal{F}_{t,n}$ . Since for each  $i = 0, 1, \dots, n$  the CPM with the monomial representation  $a^{n-i} z^i$  is concatenation of  $(n-i)!i!$  DPMs with the monomial representation  $\frac{a^{n-i} z^i}{(n-i)!i!}$ ,  $f$  can be written as

$$f = \sum_{i=0}^n \gamma_i (n-i)!i! \frac{a^{n-i} z^i}{(n-i)!i!}. \quad (35)$$

Define  $G$  by  $G = \gcd\{\gamma_i (n-i)!i! : i = 0, 1, \dots, n-t\}$ . Then, we can define  $\pi : \mathcal{F}_{t,n} \rightarrow \mathcal{G}_{t,n}^*$  as  $\pi : f \mapsto \sum_{i=0}^n \frac{\gamma_i (n-i)!i!}{G} \frac{a^{n-i} z^i}{(n-i)!i!}$ . It is easy that this  $\pi$  is surjective.  $\square$

We call the operation converting  $f \in \mathcal{F}_{t,n}$  into  $\pi(f) \in \mathcal{G}_{t,n}^*$  the *contraction*.

Notice that Theorem 2(b) guarantees that  $(Y_0, Y_1) \stackrel{\text{def}}{=} \sigma^{-1}(\pi(f))$  becomes a pair of generating matrices of the  $(t, n)$ -VSSS. Obviously, while the relative difference caused by  $(Y_0, Y_1)$  is the same as that of  $(X_0, X_1)$ , the number of rows of  $Y_i$  becomes  $1/G$  times as  $X_i$  for  $i = 0, 1$ . Summarizing, we have the following theorem giving a general formula of the  $(t, n)$ -VSSS:

**Theorem 4.** Let  $n \geq 2$  and  $2 \leq t \leq n$  be arbitrary integers. Then, for each  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}_{n-t+1}$ ,  $f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)}$  leads to a pair of the generating matrices  $(Y_0, Y_1) = \sigma^{-1}(\pi(f)) \in \mathcal{N}_{t,n}$ . The relative difference  $\alpha$  and the number of rows  $M$  of such a  $(Y_0, Y_1)$  is given by (33) and  $M = \|f\| \cdot n! / (2G)$ , respectively, where, letting  $f = \sum_{i=0}^n \gamma_i a^{n-i} z^i$  denote the expansion of  $f$ ,  $G$  is defined by  $G = \gcd\{\gamma_i (n-i)!i! : i = 0, 1, \dots, n\}$ .

*Example 1.* We construct a pair of generating matrices  $(Y_0, Y_1)$  for the  $(3, 4)$ -VSSS by using Theorem 4. Theorem 4 tells us that for each  $(\beta_0, \beta_1) \in \mathcal{B}_2$   $f = \beta_0 a(a-z)^3 + \beta_1 z(a-z)^3$  yields  $(Y_0, Y_1) \in \mathcal{N}_{3,4}$ . If we set  $(\beta_0, \beta_1) = (1, 1)$ , it easily follows that

$$\begin{aligned} f &= a(a-z)^3 + z(a-z)^3 \\ &= a^4 - 2a^3z + 2az^3 - z^4 \\ &= 4! \frac{a^4}{4!} - 2 \cdot 3! \frac{a^3z}{3!1!} + 2 \cdot 3! \frac{az^3}{1!3!} - 4! \frac{z^4}{4!} \\ &= 2 \cdot 3! \left[ 2 \frac{a^4}{4!} - \frac{a^3z}{3!1!} + \frac{az^3}{1!3!} - 2 \frac{z^4}{4!} \right], \end{aligned} \quad (36)$$

which means that  $g \stackrel{\text{def}}{=} \pi(f) = 2 \frac{a^4}{4!} - \frac{a^3z}{3!1!} + \frac{az^3}{1!3!} - 2 \frac{z^4}{4!}$ . Since the decomposition of  $g$  is given by  $g^+ = 2 \frac{a^4}{4!} + \frac{az^3}{1!3!}$  and  $g^- = \frac{a^3z}{3!1!} + 2 \frac{z^4}{4!}$ , the concatenations of DPMs corresponding to  $g^+$  and  $g^-$  become  $Y_0$  and  $Y_1$  respectively. By using



Proposition 1(a) we obtain

$$Y_0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad Y_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This pair of generating matrices, which yields  $\alpha = 1/6$ , coincides with the pair of generating matrices of the  $(3, n)$ -VSSS given by Naor and Shamir [14] with  $n = 4$ . Recall that  $(Y_0, Y_1)$  is heuristically constructed in [14].  $\square$

Notice that the converse of Theorem 4 is also true. That is, we can show that for any  $(Y_0, Y_1) \in \mathcal{N}_{t,n}$  there exists an  $f \in \mathcal{F}_{t,n}$  satisfying  $(Y_0, Y_1) = \sigma^{-1}(\pi(f))$ . This property is due to the fact that  $\pi : \mathcal{F}_{t,n} \rightarrow \mathcal{G}_{t,n}^*$  is surjective and there exists a surjection  $\tilde{\pi} : \mathcal{G}_{t,n} \rightarrow \mathcal{G}_{t,n}^*$  which is defined similarly to  $\pi$  in Proposition 2.

### 3.2 Construction of Suboptimal $(t, n)$ -VSSS Using the Formula

In this subsection we consider construction of an optimal  $(t, n)$ -VSSS by using Theorem 4. We consider the following two kinds of criteria for optimization: (A) maximization of the relative difference  $\alpha$ , and (B) minimization of the number of rows  $M$ . If there exist more than one pair of generating matrices with maximum  $\alpha$  under (A), we choose the pair with the smallest  $M$ . On the other hand, if there exist more than one pair of generating matrices with minimum  $M$  under (B), we choose the pair of with the greatest  $\alpha$ .

How can we find the optimal  $(Y_0, Y_1) \in \mathcal{N}_{t,n}$  by using Theorem 4 under criteria (A) or (B)? Unfortunately, it is quite difficult to find the optimal  $(Y_0, Y_1)$  theoretically because the formulas of  $\alpha$  and  $M$  given in Theorem 4 include  $\|f\|$  or  $G$ . However, we can use Theorem 4 in the following way for finding a suboptimal pair of generating matrices of  $(t, n)$ -VSSS. We first choose a subset  $\mathcal{B}'_{n-t+1} \subset \mathcal{B}_{n-t+1}$  with a finite number of elements adequately. Next, for each  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}'_{n-t+1}$  we expand  $f = \sum_{i=0}^{n-t} \beta_i e_{t,n}^{(i)}$  to the form  $f = \sum_{i=0}^n \gamma_i a^{n-i} z^i$  and compute  $G = \gcd\{\gamma_i(n-i)!i! : i = 0, 1, \dots, n\}$ . Since Theorem 4 tells us that both  $M$  and  $\alpha$  are determined from  $f$  and  $G$ , recalling that  $|\mathcal{B}'_{n-t+1}| < \infty$ , we can find  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}'_{n-t+1}$  that leads to  $(Y_0, Y_1) \in \mathcal{N}_{t,n}$  optimal in  $\mathcal{B}'_{n-t+1}$ . Though we mention only (A) and (B) as criteria of optimization here, such a search is possible under another criterion given in [7, 15]. In addition, notice that, since  $\mathcal{B}_{n-t+1}$  is a countably infinite set, we can choose  $\mathcal{B}'_{n-t+1}$  such that the suboptimal  $(Y_0, Y_1)$  becomes globally optimal as  $|\mathcal{B}'_{n-t+1}| \rightarrow \infty$ .

In our computer search, we defined  $\mathcal{B}'_{n-t+1}$  as the collection of all  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathbf{Z}^{n-t+1}$  satisfying  $\beta_i \geq 0$  for all  $i=0, 1, \dots, n-t$ ,  $\gcd\{\beta_0, \beta_1, \dots, \beta_{n-t}\} = 1$  and  $\sum_{i=0}^{n-t} \beta_i \leq 120$ . For each  $n \leq 9$  and  $2 \leq t \leq n-1$  we exhaustively searched for  $(\beta_0, \beta_1, \dots, \beta_{n-t}) \in \mathcal{B}'_{n-t+1}$  that yields  $(Y_0, Y_1) \in \mathcal{N}_{t,n}$  with the optimality in  $\mathcal{B}'_{n-t+1}$  under the two respective criteria. Clearly, time required for this search becomes long as  $n-t+1$  increases. However, for small  $n$  such as 9 this search

**Table 1.** Suboptimal  $(t, n)$ -VSSS in  $\mathcal{B}'_{n-t+1}$  for  $3 \leq n \leq 9$  under (A)

$(t, n)$	$M$	$\alpha$	$\{\beta_i\}_{i=0}^{n-t}$	$(t, n)$	$M$	$\alpha$	$\{\beta_i\}_{i=0}^{n-t}$
(2,3)	3	$\frac{1}{3}$	$\{1, 2\}, \{2, 1\}$	(5,7)	48	$\frac{1}{48}$	$\{2, 3, 2\}$
(2,4)	6	$\frac{1}{3}$	$\{1, 2, 1\}$	(6,7)	70	$\frac{1}{70}$	$\{3, 4\}, \{4, 3\}$
(3,4)	6	$\frac{1}{6}$	$\{1, 1\}$	(2,8)	70	$\frac{2}{7}$	$\{1, 2, 3, 4, 3, 2, 1\}$
(2,5)	10	$\frac{3}{10}$	$\{2, 4, 6, 3\},$ $\{3, 6, 4, 2\}$	(3,8)	42	$\frac{2}{21}$	$\{1, 3, 4, 4, 3, 1\}$
(3,5)	8	$\frac{1}{8}$	$\{3, 4, 3\}$	(4,8)	160	$\frac{3}{80}$	$\{9, 20, 26, 20, 9\}$
(4,5)	15	$\frac{1}{15}$	$\{2, 3\}, \{3, 2\}$	(5,8)	112	$\frac{1}{56}$	$\{2, 5, 5, 2\}$
(2,6)	20	$\frac{3}{10}$	$\{1, 2, 3, 2, 1\}$	(6,8)	198	$\frac{1}{99}$	$\{15, 26, 15\}$
(3,6)	10	$\frac{1}{10}$	$\{2, 3, 3, 2\}$	(7,8)	140	$\frac{1}{140}$	$\{1, 1\}$
(4,6)	36	$\frac{1}{18}$	$\{4, 7, 4\}$	(2,9)	126	$\frac{5}{18}$	$\{4, 8, 12, 16, 20, 15, 10, 5\},$ $\{5, 10, 15, 20, 16, 12, 8, 4\}$
(5,6)	30	$\frac{1}{30}$	$\{1, 1\}$	(3,9)	56	$\frac{5}{56}$	$\{5, 15, 21, 23, 21, 15, 5\}$
(2,7)	35	$\frac{2}{7}$	$\{3, 6, 9, 12, 8, 4\},$ $\{4, 8, 12, 9, 6, 3\}$	(4,9)	630	$\frac{2}{63}$	$\{3, 7, 10, 10, 7, 3\}$
(3,7)	30	$\frac{1}{10}$	$\{3, 9, 11, 9, 3\}$	(5,9)	8064	$\frac{13}{896}$	$\{11, 29, 37, 29, 11\}$
(4,7)	70	$\frac{3}{70}$	$\{15, 32, 38, 20\},$ $\{20, 38, 32, 15\}$	(6,9)	1764	$\frac{1}{147}$	$\{19, 39, 37, 17\}$
				(7,9)	252	$\frac{1}{252}$	$\{1, 2, 1\}$
				(8,9)	315	$\frac{1}{315}$	$\{4, 5\}, \{5, 4\}$

was completed in realistic time (at most several days) when we used a personal computer with a Pentium III 1.0GHz processor.

Table 1 shows  $M$  and  $\alpha$  of generation matrices of  $(t, n)$ -VSSS that is optimal in  $\mathcal{B}'_{n-t+1}$  under criterion (A). While [3] discusses the optimality on  $\alpha$  for  $t = 3, 4, 5, n-1$  from a combinatoric viewpoint under (A), their approach cannot be applied to the cases of  $6 \leq t \leq n-2$ . We found pairs of generating matrices of (6, 8)-, (6, 9)- and (7, 9)-VSSSs with the optimality (A) in  $\mathcal{B}'_{n-t+1}$ . For each  $2 \leq n \leq 9$  and  $2 \leq t \leq 4$ ,  $\alpha$  in Table 1 attains the theoretical upper bound given in [8] from linear programming approach (for  $t \geq 5$  no upper bound is given in [8]). In addition, for each  $2 \leq n \leq 9$  and  $2 \leq t \leq 5$   $\alpha$  in Table 1 is greater than or equal to  $\alpha$  in [5] except for the case of (5, 7)-VSSS (for  $t \geq 6$   $\alpha$  is not written in [5]). The pair of generating matrices of the (5, 7)-VSSS in [5], yielding  $\alpha = 4/147$ , may not belong to  $\mathcal{N}_{t,n}$  because we cannot find such a pair even from a larger set  $\{(\beta_0, \beta_1, \beta_2) \in \mathbf{Z}^3 : \beta_0 + \beta_1 + \beta_2 > 0, |\beta_i| \leq 1000 \text{ for } i=0,1,2\}$ . Furthermore, a pair of generating matrices of the (4, 7)-VSSS, which is written as  $g = 15\frac{a^7}{7!} - 4\frac{a^6z}{6!1!} + \frac{a^3z^4}{3!4!} - 6\frac{az^6}{1!6!} + 20\frac{z^7}{7!}$  in the polynomial expression and was first reported in [12], was turned out to be optimal in  $\mathcal{B}'_{n-t+1}$  (the method for finding such  $g$  is not written in [12]). Clearly, this pair of generating matrices, yielding  $\alpha = 3/70$  and  $M = 70$ , is better than the pair given by [3] with  $\alpha = 3/80$  and  $M = 160$ .

On the other hand, if  $n \leq 9$ , under (B) we found pairs of generating matrices  $(Y_0, Y_1)$  with the same number of rows as the pairs given by Droste [6] except

for the case of  $(6, 8)$ -VSSS. While Droste [6] mentions the existence of  $(Y_0, Y_1)$  with  $M = 128$  and  $\alpha = 1/128$ , we found  $(Y_0, Y_1)$  with  $M = 126$  and  $\alpha = 1/126$  (choose  $(\beta_0, \beta_1, \beta_2) = (5, 14, 9)$  or  $(9, 14, 5)$ ). In addition, for the case of  $n = 10$ , we found a pair of generating matrices of the  $(8, 10)$ -VSSS with  $M = 590$  and  $\alpha = 1/590$  (choose  $(\beta_0, \beta_1, \beta_2) = (14, 22, 9)$ ), though Droste [6] just mentions the existence of  $(Y_0, Y_1)$  with  $M = 640$  and  $\alpha = 1/640$ .

It is also interesting to find a  $(\beta_0, \beta_1, \dots, \beta_{n-t+1}) \in \mathcal{B}_{n-t+1}$  that lead to a simple pair of generating matrices  $(Y_0, Y_1) \in \mathcal{N}_{t,n}$ . We conclude the paper by giving such a  $(Y_0, Y_1)$  expressed in the polynomial representation for  $t = n, n-1, 2$

(i)  $(n, n)$ -VSSS

Theorem 3 tells us that elements of  $\mathcal{F}_{n,n}$  can be written as  $f = \beta_0(a-z)^n$ , where  $\beta_0$  is a positive integer. Then, it easily follows that

$$\begin{aligned} f &= \beta_0 \sum_{i=0}^n (-1)^i \binom{n}{i} a^{n-i} z^i \\ &= \beta_0 n! \left[ \sum_{i=0}^n (-1)^i \frac{a^{n-i} z^i}{(n-i)! i!} \right] \end{aligned}$$

Hence, we have  $\pi(f) = \sum_{i=0}^n (-1)^i \frac{a^{n-i} z^i}{(n-i)! i!}$  that is independent of  $\beta_0$ . This is a pair of generating matrices with  $\alpha = 1/2^n$  given by Naor and Shamir [14].

(ii)  $(n-1, n)$ -VSSS

Theorem 3 guarantees that  $f \in \mathcal{F}_{n-1,n}$  can be expressed as  $f = \beta_0 a(a-z)^{n-1} + \beta_1 z(a-z)^{n-1}$ , where  $(\beta_0, \beta_1) \in \mathcal{B}_2$ . We set  $\beta_0 = \beta_1 = 1$  for even  $n$  and  $\beta_0 = \lfloor \frac{n}{2} \rfloor$  and  $\beta_1 = \lceil \frac{n}{2} \rceil$  for odd  $n$ . Then,  $\pi(f)$  can be expressed as

$$\pi(f) = \begin{cases} \sum_{i=0}^n (-1)^i \left( \frac{n}{2} - i \right) \frac{a^{n-i} z^i}{(n-i)! i!}, & \text{if } n \text{ is even,} \\ \sum_{i=0}^n (-1)^i \left( \frac{n+1}{2} - i \right) \frac{a^{n-i} z^i}{(n-i)! i!}, & \text{if } n \text{ is odd,} \end{cases}$$

which leads to the pair of generating matrices with  $M = \frac{n}{2} \binom{n-1}{n/2-1}$  and  $\alpha = 1/\lfloor \frac{n}{2} \binom{n-1}{n/2-1} \rfloor$  for even  $n$  and  $M = n \binom{n-1}{(n-1)/2}$  and  $\alpha = 4/\lfloor n \binom{n-1}{(n-1)/2} \rfloor$  for odd  $n$ . These pairs of generating matrices are given by Blundo et al [3].

(iii)  $(2, n)$ -VSSS

By Theorem 3,  $f \in \mathcal{F}_{2,n}$  can be written as  $f = \sum_{i=0}^{n-2} \beta_i a^{n-2-i} z^i (a-z)^2$ , where  $(\beta_0, \beta_1, \dots, \beta_{n-2}) \in \mathcal{B}_{n-1}$ . If we set  $\beta_i = n-1-i$  for all  $i = 0, 1, \dots, n-2$ , we have

$$\pi(f) = (n-1) \frac{a^n}{n!} - \frac{a^{n-1} z}{(n-1)! 1!} + \frac{z^n}{n!},$$

which leads to the pair of generating matrices given by Naor and Shamir [14] with  $M = n$  and  $\alpha = 1/n$ . On the other hand, for the case of even  $n$  if we set

$\beta_i = i + 1$  for  $i = 0, 1, \dots, \frac{n}{2} - 1$  and  $\beta_i = n - i - 1$  for  $i = \frac{n}{2}, \frac{n}{2} + 1, \dots, n - 2$ , we have

$$\pi(f) = \frac{1}{2} \binom{n}{n/2} \frac{a^n}{n!} - \frac{a^{n/2} z^{n/2}}{(n/2)!(n/2)!} + \frac{1}{2} \binom{n}{n/2} \frac{z^n}{n!},$$

The pair of generating matrices corresponding to  $f$  above satisfies  $M = \binom{n}{n/2}$  and  $\alpha = \frac{n}{4(n-1)}$ .

## Appendix

### A Proof of Theorem 1

We prove Theorem 1(a) here because Theorem 1(b) can be developed similarly. Let  $X_0$  and  $X_1$  be concatenations of CPMs with the polynomial representations  $f_0$  and  $f_1$ , respectively. By the assumption of the theorem,  $f_0$  and  $f_1$  satisfy (7) and (8). In view of Definition 1 and the definition of the generating matrices, it is sufficient to prove that (i)  $X_0[S] = X_1[S]$  for any  $S \in 2^P$  with  $|S| = t - 1$ , and (ii)  $h(\text{OR}(X_0[S])) < h(\text{OR}(X_1[S]))$  for any  $S \in 2^P$  with  $|S| = t$ .

The proof of property (i) is simple. Since Proposition 1 tells us that for  $i = 0, 1$  application of  $\psi^{n-t+1}$  to  $f_i$  means elimination of arbitrary  $n - t + 1$  rows from  $X_i$ , (7) implies that  $X_0[S] = X_1[S]$  for any  $S \in 2^P$  with  $|S| = t - 1$ . This establishes property (i). On the other hand, we notice that  $\psi^{n-t} f_i|_{z=0}$  means the number of the CPMs represented as  $a^t$  in  $\text{OR}(X_i[S])$  for any  $S \in 2^P$  with  $|S| = t$ . Then, (8) implies that  $\text{OR}(X_0[S])$  contains the CPMs represented as  $a^t$  more than  $\text{OR}(X_1[S])$ , which immediately leads to  $h(\text{OR}(X_0[S])) < h(\text{OR}(X_1[S]))$ . This establishes property (ii).

### B Proof of Lemma 2

We prove Lemma 2 by induction on  $l$ . The claim of the lemma is trivial if  $l = 0$ . Let  $l \geq 1$  be an arbitrary integer and suppose that an arbitrary  $g \in \mathcal{R}_n$  with  $g = z^l h$  for some  $h \in \mathcal{R}_{n-l}$  satisfies  $\psi^l g = 0$ . Since

$$\begin{aligned} \psi^l g &= \psi^l(z^l h) \\ &= \psi^{l-1}(\{lh + z \cdot (\psi h)\}z^{l-1}), \end{aligned} \tag{B.1}$$

we have

$$lh + z \cdot (\psi h) = 0 \tag{B.2}$$

by induction hypothesis. Setting

$$h = \gamma_0 a^{n-l} + \gamma_1 a^{n-l-1} z + \dots + \gamma_{n-l} z^{n-l},$$

(B.2) leads to

$$l\gamma_0 a^{n-l} + [(n-l)\gamma_0 + 2\gamma_1]a^{n-l-1}z + \dots + [\gamma_{n-l-1} + (n-l+1)\gamma_{n-l}]z^{n-l} = 0, \tag{B.3}$$

which mean that  $l\gamma_0 = (n-l)\gamma_0 + 2\gamma_1 = \dots = \gamma_{n-l-1} + (n-l+1)\gamma_{n-l} = 0$  and therefore  $\gamma_0 = \gamma_1 = \dots = \gamma_{n-l} = 0$ .

## References

1. C. Blundo and A. De Santis, "Visual cryptography schemes with perfect reconstruction of black pixels," *Computer and Graphics*, vol. 22, no. 4, pp. 449-455, 1998.
2. C. Blundo, A. De Santis and D. R. Stinson, "On the contrast in visual cryptography schemes," *Journal of Cryptology*, vol. 12, no. 4, pp. 261-289, 1999.
3. C. Blundo, P. D'Arco, A. De Santis and D. R. Stinson, "Contrast optimal threshold visual cryptography schemes," submitted to *SIAM Journal on Discrete Mathematics*. (Available from <http://cacr.math.uwaterloo.ca/~dstinson>)
4. C. Blundo, A. De Bonis, and A. De Santis, "Improved schemes for visual cryptography," *Designs, Codes, and Cryptography*, vol. 24, pp. 255-278, 2001.
5. C. K. Choi, S. S. Yang, J. H. Park and R. Kohno, "New construction for improving contrast in visual cryptography," *Proc. of ISITA*, Mexico City, pp. 368-371, 1998.
6. S. Droste, "New results on visual cryptography," *Advance in Cryptography-CRYPT'96*, LNCS 1109, pp. 401-415, Springer Verlag, 1996.
7. P. A. Eisen and D. R. Stinson, "Threshold visual cryptography scheme with specified whiteness levels of reconstructed pixels," *Designs, Codes, and Cryptology*, vol. 25, No. 1, pp. 15-61, 2002.
8. T. Hofmeister, M. Krause and H. U. Simon, "Contrast-optimal  $k$  out of  $n$  secret sharing schemes in visual cryptography," *Theoretical Computer Science*, vol. 240, pp. 471-485, 2000.
9. T. Kato and H. Imai, "An extended construction method of visual secret sharing scheme," *IEICE Trans.*, vol. J79-A, no. 8, pp. 1344-1351, 1996. (in Japanese)
10. H. Koga, M. Iwamoto and H. Yamamoto, "An analytic construction of the visual secret sharing scheme for color images," *IEICE Trans. on Fundamentals*, vol. E84-A, no. 1, pp. 262-272, 2001.
11. M. Krause and H. U. Simon, "Determining the optimal contrast for secret sharing in visual cryptography," *Latin 2000*, LNCS 1776, pp. 280-291, 2000.
12. H. Kuwakado and H. Tanaka, "Polynomial representation of visual secret sharing scheme for black-white images," *Proc. of 2001 Symposium on Cryptography and Information Security*, pp. 417-422, 2001.
13. H. Kuwakado and H. Tanaka, "Polynomial representation of visual secret sharing scheme and its application," *IEICE Trans. on Fundamentals*, vol. E85-A, no. 6, pp. 1379-1386, 2002.
14. M. Naor and A. Shamir, "Visual cryptography," *Advance in Cryptography-EUROCRYPT'94*, LNCS 950, pp. 1-12, Springer-Verlag, 1994.
15. E. R. Verheul and H. C. A. van Tilborg, "Constructions and properties of  $k$  out of  $n$  visual secret sharing schemes," *Designs, Codes, and Cryptography*, vol. 11, no. 2, pp. 179-196, 1997.

# On Unconditionally Secure Robust Distributed Key Distribution Centers

Paolo D'Arco<sup>1</sup> and Douglas R. Stinson<sup>2</sup>

<sup>1</sup> Department of Combinatorics and Optimization, University of Waterloo,  
N2L 3G1, Waterloo Ontario, Canada,  
[pdarco@cacr.math.uwaterloo.ca](mailto:pdarco@cacr.math.uwaterloo.ca)

<sup>2</sup> School of Computer Science, University of Waterloo,  
N2L 3G1, Waterloo Ontario, Canada,  
[dstinson@cacr.math.uwaterloo.ca](mailto:dstinson@cacr.math.uwaterloo.ca)

**Abstract.** A Key Distribution Center enables secure communications among groups of users in a network by providing common keys that can be used with a symmetric encryption algorithm to encrypt and decrypt messages the users wish to send to each other. A Distributed Key Distribution Center is a set of servers of a network that *jointly* realize a Key Distribution Center. In this paper we propose an unconditionally secure scheme to set up a *robust* Distributed Key Distribution Center. Such a distributed center keeps working even if some minority of the servers malfunction or misbehave under the control of a *mobile* adversary. Our scheme for a distributed key distribution center is constructed using unconditionally secure proactive verifiable secret sharing schemes. We review the unconditionally secure verifiable secret sharing scheme described by Stinson and Wei, discuss a problem with the proactive version of that scheme, and present a modified version which is proactively secure.

## 1 Introduction

A group of users of a network, referred to as a “conference”, in order to securely communicate over public channels, could decide to use symmetric encryption algorithms, e.g., RC6 or AES. These algorithms are fast and presumed to be secure. But to apply this strategy, they need a common key with which to encrypt and to decrypt the messages they wish to send to each other. This basic problem is well-known in the literature and it is called the *Key Establishment Problem*.

A common solution to the Key Establishment Problem is to use a Key Distribution Center (KDC, for short), in which a server is responsible for the distribution and management of the secret keys. The idea is the following: Each user shares a common key with the center. When he wants to securely communicate with a subset of other users, he sends a request for a conference key. The center checks for membership of the user in that conference, and generates and distributes the conference key in encrypted form to each member of the group. Needham and Schroeder [13] initiated this approach, implemented most notably

in the Kerberos System [14]. Kerberos was formally defined and studied in [1], where it is referred to as the *three-party model*.

The scheme implemented by the Key Distribution Center is called a *Key Distribution Scheme* (KDS, for short). The scheme is said to be *unconditionally secure* if it is secure independent of the computational resources of the adversary. Several kinds of Key Distribution Schemes have been considered in the literature: Key Pre-Distribution Schemes (KPS, for short), Key Agreement Schemes (KAS, for short) and Broadcast Encryption Schemes (BES, for short) among others. The reader can consult [20] for a survey on unconditionally secure schemes, [19,11] for a general and detailed description of a variety of protocols for the Key Establishment Problem and related issues, and [3] for a simple introduction.

Our attention in this paper focuses on a model which remedies some potential weaknesses introduced by using a *single KDC*. Indeed, the main drawback of a single KDC is that it must be *trusted*. Potentially, it could eavesdrop on all the communications. Moreover, the center can be a “bottleneck” for the performance of the network and, if it crashes, secure communication cannot be supported anymore. Last but not least, even if the KDC is honest and everything works fine, the KDC still represents an attractive target to the adversary because the overall system security is lost if the KDC is compromised.

In order to solve the above problems, a new approach to key distribution was introduced in [12]. A Distributed Key Distribution Center (DKDC, for short) is a set of  $n$  servers of a network that *jointly realize* the function of a Key Distribution Center. A user who needs to participate in a conference sends a key-request to a subset of his own choosing of the  $n$  servers, and the contacted servers answer with some information enabling the user to compute the conference key. In such a model, a single server by itself does not know the secret keys, since they are *shared* between the  $n$  servers. Moreover, if some server crashes, secure communication can still be supported by the other servers and, since each user can contact a different subset of servers, the slow-down factor for the performance of the applications introduced by a single KDC can be improved.

In subsequent papers [6,24], the notion of DKDC has been studied from an information theoretic point of view. Therein, the authors showed that the protocol proposed in [12], based on bivariate polynomials, is optimal with respect to the amount of information needed to set up and manage the system.

In this paper we show how to set up a Robust DKDC. Namely, we describe a protocol where each server can *verify* that the information it stores and uses to answer the user’s key-request messages is consistent with the information stored by the other servers; at the same time, the users are *guaranteed* that they can compute the same key for a given conference in which they belong to. Moreover, time is divided in *periods*, and at the beginning of each period the servers are involved in an update procedure that “refreshes” the private information they store while the conference keys they provide stay the same. This property is referred to as *proactive security*. Notice that, in [12], a simple solution was outlined, which could have been applied to the basic polynomial

construction they proposed in order to achieve the above properties. However, as we show here, that solution does not work.

The design of our DKDC is based on unconditionally secure proactive verifiable secret sharing. In Section 5 we show that some existing schemes [21,15] contain flaws. Then, we describe two techniques to modify these schemes in order to realize the proactive security property. We point out that the same ideas can be used to provide the proactive security property even to the verifiable secret sharing schemes given in [8].

## 2 The Model

Let  $\mathcal{U} = \{U_1, \dots, U_m\}$  be a set of  $m$  users and let  $\mathcal{S} = \{S_1, \dots, S_n\}$  be a set of  $n$  servers. Each user has *secure channels* connecting him or her to *all* the servers. Each pair of servers is connected by a *secure channel* and all of them share a *broadcast channel* and a *global clock* (i.e., the system is synchronous). Servers can be good (i.e., they honestly execute the protocol) or bad (i.e., they are controlled by an adversary and can deviate from the protocol in arbitrary ways) but a majority of good servers is always present across the system. Let  $\mathcal{C}$  be the set of *conferences*, i.e., the set of groups of users which want to securely communicate, and let  $\mathcal{G}$  be the set of *tolerated coalitions*, i.e., the set of coalitions of users who can try to break the scheme in some way. For example,  $\mathcal{C}$  could be the set of all subsets of users of size  $p$  while  $\mathcal{G}$  could be the set of all subsets of users of size  $q$ . A *verifiable distributed key distribution scheme* is divided in three phases: an *initialization phase*, which involves only the servers; a *key-request phase*, in which users ask servers for keys; and a *key-computation phase*, in which users construct keys from the messages they received from the servers who were contacted during the key-request phase.

*Initialization Phase.* We assume that the initialization phase is performed by a *joint computation* of all the servers. Each of them, using a *private source* of randomness,  $r_i$ , generates some messages that it securely sends to the others. More precisely, for  $i = 1, \dots, n$ ,  $S_i$  sends to  $S_j$  some message  $\gamma_{i,j}$ , for each  $j = 1, \dots, n$ . At the end of the distribution phase, for  $i = 1, \dots, n$ , each server  $S_i$  *verifies* the information received, sends messages along the broadcast channel and, eventually, *computes and stores* some secret information  $a_i = f(\gamma_{1,i}, \dots, \gamma_{n,i})$ , where  $f$  is a publicly known function. Moreover, each server constructs a list  $\mathcal{L}$  of the good servers present across the network at the end of this phase (the lists held by the good servers will all contain the same identifiers).

*Key-Request Phase.* Let  $C_h \in \mathcal{C}$  be a conference. Each user  $U_j$  in  $C_h$  contacts a subset of a certain size of good servers belonging to  $\mathcal{L}$ , requesting a key for the conference  $C_h$ . We denote this key by  $\kappa_h$ . Each good server  $S_i$ , contacted by user  $U_j$ , checks<sup>1</sup> for membership of  $U_j$  in  $C_h$ ; if  $U_j \in C_h$ , then  $S_i$  computes

<sup>1</sup> We do not consider the underlying authentication mechanism involved in a key request phase.



a value  $y_{i,j}^h = F(a_i, j, h)$ , where  $F$  is a publicly known function. Otherwise,  $S_i$  sets  $y_{i,j}^h = \perp$  (a special value which conveys no information about  $\kappa_h$ ). Finally,  $S_i$  sends the value  $y_{i,j}^h$  to  $U_j$ . Note that a bad server can either refuse to reply or it may send some incorrect value.

*Key-Computation Phase.* Having received the values from the servers, each user  $U_j$  in  $C_h$  computes  $\kappa_h$  from a certain majority of the values received.

Roughly speaking, a Verifiable DKDC must satisfy the following properties:

- **Correct and Verifiable Initialization Phase.** When the initialization phase successfully terminates, any good server  $S_i$  must be able to identify the subset of good servers and to compute his private information  $a_i$ .
- **Consistent Key Computation.** Each user in a conference  $C_h \subseteq \mathcal{U}$  must be able to compute *the same* conference key, after interacting with a subset of good servers of a certain size.
- **Conference Key Security.** A conference key must be secure against attacks performed by coalitions of bad servers, coalitions of users, and hybrid coalitions of a certain size consisting of servers and users.

Let  $b$  be an upper bound on the number of bad servers during any phase of the protocol, and let  $t > b$  denote a sufficient number of parties to reconstruct a possible conference key. In a more precise way, we state the following definition:

**Definition 1.** Let  $b, t$  and  $n$  be integers such that  $n \geq t$  and  $t > b$ . Let  $\mathcal{U} = \{U_1, \dots, U_m\}$  be a set of  $m$  users, and let  $\mathcal{S} = \{S_1, \dots, S_n\}$  be a set of  $n$  servers. Finally, let  $\mathcal{C} \subseteq 2^{\mathcal{U}}$  be the set of conferences and let  $\mathcal{G} \subseteq 2^{\mathcal{U}}$  be the set of tolerated coalitions. A verifiable  $(b, t, n, \mathcal{C}, \mathcal{G})$ -Distributed Key Distribution Scheme (for short,  $(b, t, n, \mathcal{C}, \mathcal{G})$ -VDKDS) is a three-phase protocol consisting of an Initialization Phase, a Key Request Phase, and a Key Computation Phase, which enables each user in  $C_h \in \mathcal{C}$  to compute a common key  $\kappa_h$  by interacting with at least  $n - b$  servers of the network. More precisely, the following properties are satisfied:

1. After the initialization phase, each good server computes his private information and verifies its consistency with the information received and stored by the other good servers. At least  $n - b$  servers successfully complete this phase and each of them construct the same (public) list  $\mathcal{L}$  containing the identities of the good servers.
2. Each user in  $C_h \in \mathcal{C}$  can compute the common key  $\kappa_h$  by contacting the servers in  $\mathcal{L}$ . At least  $|\mathcal{L}| - b > t$  of the  $|\mathcal{L}|$  servers give good answers, from which the user reconstructs the key. Any  $t$  good answers are sufficient to reconstruct the key.
3. Each conference key is completely secure against coalitions of users  $G \in \mathcal{G}$ ; coalitions of servers of size less than  $b$ ; and joint coalitions of at most  $b$  servers and users in a subset  $G \in \mathcal{G}$ .

Basically, in the above model, we assume that at most  $b$  servers can misbehave during the initialization phase of the system, and during the key request phase. Moreover, these two subsets of bad servers can be different: in other words, we assume that the adversary is *mobile*. Notice that a crash of some of the servers in the above model can be seen as a simple type of misbehavior.

### 3 A Verifiable Secret Sharing Scheme

The main component of our  $(b, t, n, \mathcal{C}, \mathcal{G})$ -VDKDS is a Verifiable Secret Sharing Scheme (VSS, for short). Loosely speaking, in a VSS Scheme, a Dealer shares a secret among a set of participants in such a way that each participant can verify if the shares he gets, from the dealer during the distribution phase and from the other participants during the recovering phase, are consistent with the secret. VSS schemes were introduced in [5].

In this section we describe the VSS we are going to use. It is a slightly modified version of the scheme proposed by Stinson and Wei in [21], whose round complexity has been improved by applying the technique recently described in [8]. Due to the use of a symmetric polynomial, the scheme of [21], enhanced with the ideas of [8], is a bit more efficient than the scheme described in [8] with the same parameters (i.e., when  $b < \frac{n}{4}$ ). Notice that, in the following construction, the dealer, after sending messages during the initialization phase, becomes inactive. In fact, as we will argue later, he can be completely substituted by a joint computation performed by the servers of the system.

First of all, we recall the definition of a VSS.

**Definition 2.** Let  $\mathcal{D}$  be a dealer and let  $P_1, \dots, P_n$  be  $n$  participants connected by secure channels and having access to a broadcast channel. Moreover, let  $\mathcal{A}$  be an adversary that can corrupt up to  $b$  of the participants (including the dealer). Assume that  $\pi$  is a protocol consisting of two phases, Share and Reconstruct, and let  $S$  be a set of possible secret values. At the beginning of Share, the dealer inputs a secret  $s \in S$ . At the end of Share each participant  $P_i$  outputs a boolean value  $ver_i$ . At the end of Reconstruct each participant outputs a value in  $S$ . The protocol  $\pi$  is an  $(n, t, b, S)$  Unconditionally Secure Verifiable Secret Sharing Scheme if the following properties are satisfied:

1. If a good player  $P_i$  outputs  $ver_i = 0$  at the end of Share, then every good player outputs  $ver_i = 0$ .
2. If the dealer is good, then  $ver_i = 1$  for every good  $P_i$ .
3. If at least  $n - b$  players  $P_i$  output  $ver_i = 1$  at the end of Share, then there exists an  $s' \in S$  such that the event that all good  $P_i$  output  $s'$  at the end of Reconstruct is fixed at the end of Share and  $s' = s$  if the dealer is good.
4. If  $|S| = q$ ,  $s$  is chosen randomly from  $S$  and the dealer is good, then any coalition of at most  $t - 1$  participants cannot guess, at the end of Share, the value  $s$  with probability greater than  $\frac{1}{q}$ .

The scheme we are going to use works as follows: let  $t, b$  be two integers such that  $n \geq t + 3b$  and  $t > b$ . Let  $S = GF(q)$  be a finite field and let  $\omega$  be a primitive element in  $GF(q)$ . All computations are done in the field  $GF(q)$ .

*Share.*

- When  $\mathcal{D}$  wants to share a secret value  $s \in S$ , he chooses a random symmetric polynomial

$$f(x, y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} a_{ij} x^i y^j,$$

where  $a_{00} = s$  and  $a_{ij} = a_{ji}$  for all  $i, j$ . Then, for each  $k$ ,  $\mathcal{D}$  sends  $h_k(x) = f(x, \omega^k)$  to  $P_k$  through a secure channel. At the same time, for each  $i$ ,  $P_i$  generates and sends to every  $P_k$  a random value  $r_{ik} \in GF(q)$  through a secure channel.

- After receiving  $h_k(x)$  from  $\mathcal{D}$  and  $r_{1k}, \dots, r_{nk}$  from the other participants, each  $P_k$  broadcasts the value  $h_k(\omega^\ell) + r_{k\ell} + r_{\ell k}$ , for each  $\ell \neq k$ .
- Each  $P_i$  computes the maximum subset  $G \subseteq \{1, \dots, n\}$  such that any ordered pair  $(\ell, k) \in G \times G$  is *consistent*, i.e. such that  $h_k(\omega^\ell) + r_{k\ell} + r_{\ell k} = h_\ell(\omega^k) + r_{\ell k} + r_{k\ell}$ . If  $|G| \geq n - b$ , then  $P_i$  outputs  $ver_i = 1$ . Otherwise,  $P_i$  outputs  $ver_i = 0$ .

*Reconstruct.*

- Each  $P_i$  sends  $h_i(0)$  to each  $P_k$ , where  $i \in G$ , the set of good participants after *Share*.
- After receiving the  $h_i(0)$ 's,  $P_k$  computes a polynomial  $f_k(0, y)$  such that  $f_k(0, \omega^i) = h_i(0)$  for at least  $n - 2b$  of the data he has received. This operation can be done efficiently, for example, either using the methods described in [18], or using error correction techniques for Reed-Solomon Codes [10].
- $P_k$  computes and outputs  $s' = f_k(0, 0)$ .

The security of the protocol can be shown along the same line of Theorem 2 in [21]. Our only change in the protocol is to *Share* where, instead of using the secure channels for the check of consistency of the shares the dealer  $\mathcal{D}$  distributes, we use random one-time pads and the broadcast channel as in [8]. With this trick we save one round of communication, compared to [21].

## 4 A Verifiable Distributed Key Distribution Scheme

Using the VSS described in the previous section, we describe a simplified version of a Verifiable DKDS. We assume that a Dealer  $\mathcal{D}$  initializes the system but, as we will show later, this assumption can be easily removed. Our scheme provides  $\ell$ -wise independent conference keys, i.e., the  $\ell$ -th conference key is uniformly distributed over the set of possible values, even if an adversary already knows  $\ell - 1$  previous conference keys. It works as follows:

*Set Up Phase.*

- Let  $\ell_G$  be the maximum number of conference keys that a group  $G$  can compute. Assume that  $\ell > \max_{G \in \mathcal{G}} \ell_G$ . The dealer chooses a *random* polynomial  $K(x) = \sum_{z=0}^{\ell-1} k_z x^z$ . The conference key for  $C_s$  is defined by  $\kappa_s = K(s)$ .
- Then, for each coefficient  $k_z$  of  $K(x)$  the dealer runs  $\ell$  independent copies  $\Sigma_z$  of the VSS described before, where the secret that  $\Sigma_z$  distributes among the servers is  $k_z$ .
- Each server  $S_i$  stores the  $\ell$  univariate polynomials  $h_i^{k_0}(x), \dots, h_i^{k_{\ell-1}}(x)$  sent by the dealer during the executions of the Share Phase of the  $\Sigma_z$ 's, and publishes the list of good servers he has found.

In a VSS, the reconstruction of the secret is done by the participants (i.e., the servers in our setting) while in a DKDS each user of a given conference contacts the servers, receives some information and computes the common key by applying a public function to the values received. A straightforward “solution” to this different scenario could be that each server  $S_i$  sends, according to the VSS scheme, the values of his polynomials evaluated in zero, i.e.,  $h_i^{k_0}(0), \dots, h_i^{k_{\ell-1}}(0)$ , to the users. But this is insecure, because, in this case, the user reconstructs *all* the keys! Thus, we need a different approach. Basically, the values sent by the servers must enable them to compute a single key, namely, the one the user is asking for.

*Key Request and Key Computation Phases.*

- User  $U_j \in C_s$  asks a subset of good servers of size at least  $n - b$  for the key  $\kappa_s$ .
- Each server  $S_i$  computes

$$\bar{h}_i(0) = h_i^{k_0}(0) + h_i^{k_1}(0)s + \dots + h_i^{k_{\ell-1}}(0)s^{\ell-1},$$

and sends  $\bar{h}_i(0)$  to the user.

- The user interpolates a polynomial  $\bar{h}(x)$  such that  $\bar{h}(\omega^i) = \bar{h}_i(0)$  for at least  $n - 2b$  of the values received. Then, he recovers  $\kappa_s = \bar{h}(0)$ .

**Correctness.** The correctness of the construction can be shown as follows: according to the VSS scheme described in the previous section, each coefficient of  $K(x)$  can be recovered by applying the Lagrange formula. More precisely, assuming that the first  $t$  servers are good servers, we have

$$k_j = \sum_{i=1}^t h_i^{k_j}(0) b_i, \quad \text{where} \quad b_i = \prod_{k \neq i} \frac{\omega^k}{\omega^k - \omega^i}.$$

Notice that,

$$\begin{aligned} K(s) &= k_0 + k_1s + \cdots + k_{\ell-1}s^{\ell-1} \\ &= \sum_{i=1}^t h_i^{k_0}(0)b_i + \left(\sum_{i=1}^t h_i^{k_1}(0)b_i\right)s + \cdots + \left(\sum_{i=1}^t h_i^{k_{\ell-1}}(0)b_i\right)s^{\ell-1} \\ &= \sum_{i=1}^t (h_i^{k_0}(0) + h_i^{k_1}(0)s + \cdots + h_i^{k_{\ell-1}}(0)s^{\ell-1})b_i = \sum_{i=1}^t \bar{h}_i(0)b_i = \bar{h}(0) = \kappa_s. \end{aligned}$$

In general, since the user does not know *a priori* which servers send correct values, he needs to interpolate a polynomial  $\bar{h}(x)$  which agrees with at least  $n - 2b$  of the values received, which can be done efficiently by applying the techniques given in [18] or in [10], exactly as in the VSS. Finally, he recovers the common key by evaluating  $\bar{h}(x)$  at  $x = 0$ .

*A One-Time Scheme (Toy Example).* In order to give to the reader a concrete idea of the protocol, let us consider the following example: let  $q = 7$ ,  $\omega = 3$ ,  $n = 5$ ,  $t = 2$  and  $b = 1$ . The dealer defines the keys as points belonging to  $K(x) = 3 + 5x(\text{mod}7)$  and, to share the coefficients 3 and 5, he chooses two symmetric bivariate polynomials, say

$$f^1(x, y) = 3 + 5x + 5y + 3xy \quad \text{and} \quad f^2(x, y) = 5 + 4x + 4y + 4xy.$$

Therefore server  $S_i$ , whose public identity is defined by  $\omega^i$ , gets two polynomials  $h_i^1(x) = f^1(x, \omega^i)$  and  $h_i^2(x) = f^2(x, \omega^i)$ . More precisely, the polynomial distributed are listed in the following table

Server	identifier	$h_i^1(x)$	$h_i^2(x)$
$S_1$	$\omega^1 = 3$	4	$3 + 2x$
$S_2$	$\omega^2 = 2$	$6 + 4x$	$6 + 5x$
$S_3$	$\omega^3 = 6$	$5 + 2x$	1
$S_4$	$\omega^4 = 4$	$2 + 3x$	$4x$
$S_5$	$\omega^5 = 5$	$6x$	$4 + 3x$

The value of the conference key  $\kappa_3 = 3 + 5 \times 3 \text{ mod } 7 = 4$ . Assume that servers  $S_1$  and  $S_2$ , belonging to the list  $\mathcal{L}$  of good servers, send to a user in  $C_3$  correct values in order to enable him to recover  $\kappa_3$ . More precisely, the user gets from  $S_1$  the value  $4 + 3 \times 3 = 6$  and the value  $6 + 6 \times 3 = 3$  from  $S_2$ . Using the public identifiers of  $S_1$  and  $S_2$ , the user sets up the two pairs of values  $(3, 6), (2, 3)$ , and by applying the Lagrange Formula, he interpolates the polynomial  $P(x) = 3 \times x - 3 \text{ mod } 7$ . It is easy to see that  $P(0) = 4$ , and hence the user recovers  $\kappa_3$ . Moreover, assuming that  $S_5$  was bad in the set up phase, the user gets from the other “supposed to be good” servers  $S_3$  and  $S_4$  the values  $2 + 0 \times 3 = 2$  and  $0 + 4 \times 3 = 5$  (if they are honest). These values belong to the polynomial interpolated. Notice that, assuming that  $S_1$  and  $S_2$  send correct values (i.e., are honest) and since at most one server (i.e.,  $b = 1$ ) can send an

incorrect value during the key request phase, at least one of the values send by  $S_3$  and  $S_4$  must agree with  $P(x)$ .

*Security.* The security of the protocol can be shown by considering the following possible cases:

- *Coalition of Users.* As long as a group  $G \in \mathcal{G}$  does not recover more than  $\ell$  conference keys and, more precisely, does not obtain information from the servers for more than  $\ell$  conference keys, the group cannot compute any information about another conference key in an information theoretic sense. This property easily follows from the assumption that the conference keys are values of a polynomial of degree  $\ell - 1$  (i.e., they are  $\ell$ -wise independent). By the  $\ell$ -wise independence, it is easy to see that a coalition holding  $\ell - 1$  pairs  $(s, \kappa_s)$ , for any choice of an  $\ell$ -th pair  $(s', \kappa_{s'})$  can interpolate a *different polynomial* of degree  $\ell - 1$ . Hence, the  $\ell$ -th key is unconditionally secure.
- *Coalition of Servers.* By the property of the VSS, any coalition of  $b$  servers, even putting together all the information received during the set up phase, cannot compute any information about any conference key, because each coefficient of the polynomial determining the keys is shared in the VSS by a  $t$ -degree polynomial, where  $t > b$ . Moreover, users reconstruct the conference keys even if at most  $b$  servers are bad during the initialization phase and at most  $b$  (possibly different) servers send incorrect information to the users during the key-request phase. Hence, in this case, the security follows from the security of the VSS (see, e.g., Theorem 2 in [21]).
- *Coalitions of Users and Servers.* The worst scenario we have to consider is when  $b$  servers collude with a group of users  $G \in \mathcal{G}$  who has run the protocol many times, recovering a bunch of conference keys. For example, assuming that the bad servers are  $S_1, \dots, S_b$ , the information the coalition possesses is given by the partial polynomials  $h_1^{k_0}(x), \dots, h_1^{k_{\ell-1}}(x), \dots, h_b^{k_0}(x), \dots, h_b^{k_{\ell-1}}(x)$  plus the values received by the users during the previous executions of the protocol in order to retrieve some conference keys. As the previous cases have shown, the two types of information by themselves are useless in order to find out information about a new key. However, it is not difficult to see that even the *joint* knowledge of this information does not help, since the coalition does not have “enough points” to interpolate a new key. Actually, any new key can still assume any possible value, for each choice of the values that should be provided by a group of at least  $t - b$  other servers (i.e., perfectly secure).

*Remark.* The Dealer  $\mathcal{D}$  can be easily removed from the above protocol since it can be removed from the VSS scheme, as shown in [21]: each participant, during *Share*, chooses a different secret value and executes the protocol. The *real* shared value is given by the *sum* of the values chosen by the good participants. Along the same line, each server of the system, during the initialization phase of the VDKDS can act as the dealer, choosing a different polynomial. In this case *the keys are points of the polynomial obtained by summing up the polynomials chosen by the good servers*. The presence of the dealer has been used only to simplify the description of the protocol. Moreover notice that the assumption that there

are at most  $b$  bad servers in this scenario implies that the Share Phase of the VSS protocol is *always successfully* completed by the good servers.

## 5 Proactivity

The concept of *proactive security* was introduced in [17] and applied to the secret sharing setting in [9]. Basically the idea is that, if the information stored by the servers in order to share a given secret *stays the same* for all the lifetime of the system, then an adversary can eventually break into a sufficient number of servers to learn or destroy the secret. On the other hand, if time is divided into *periods*, and at the beginning of each period the information stored by the servers in a given time period changes (while the shared secret stays the same), then the adversary probably does not have enough time to break into the necessary number of servers. Moreover, the information he learns during period  $p$  is useless during period  $p+i$ , for  $i = 1, 2, \dots$ . So, he has to start a new attack from scratch during each time period.

The design of a Proactive VDKDS easily follows once we have a Proactive VSS. Therefore, in the following subsections we address the construction of unconditionally secure proactive VSS. Notice that, in [12], a simple solution to set up a Proactive VDKDS was given, but as we show in Appendix A, it does not work.

### 5.1 An Unconditionally Secure Proactive VSS for $b \leq \frac{n}{4} - 1$ Bad Servers

The first unconditionally secure proactive VSS was proposed by Stinson and Wei in [21], where proactivity is added to the basic VSS described before. A generalization of that scheme has subsequently been given in [15]. We start by analyzing a weakness of the scheme given in [21], and we show how it can be used to attack the proactive security property. Then, we show a variation of the scheme that solves the problem. Moreover, we describe another technique that can be used to add proactive security to both VSSs given in [21] and [8] for the case in which the number of bad servers is  $b \leq \frac{n}{4} - 1$ .

Let  $t > b + 1$ . We assume that time is divided in periods  $p = 1, 2, \dots$ . Each good server, at the beginning of the new period, performs the steps given in the table of the next page to renew the shares [21]. Unfortunately, the symmetry of the polynomial  $r^\ell(x, y)$  can be used by bad servers to break the scheme. Indeed, during step 2, server  $P_\ell$  broadcasts the polynomial  $h_0^\ell(x) = r^\ell(x, 0) = r^\ell(0, y)$ . Hence, any server can compute the values  $h_0^\ell(\omega^k) = r^\ell(0, \omega^k) = h_k^\ell(0)$  for  $k = 1, \dots, n$ . Then, in step 6, each good player  $P_m$  updates his own share  $h_m(x)$  by adding the  $\sum_{k \in \mathcal{L}} h_m^k(x)$ . At this point notice that, according to the VSS, the only part of the share  $h_m(x)$  used to reconstruct the secret is  $h_m(0)$ , the first coefficient of the polynomial, which is updated by  $\sum_{k \in \mathcal{L}} h_m^k(0)$ .

But this sum can be computed by everybody using the public broadcasts in step 2. The consequence is that if a *passive adversary* breaks into server  $P_m$

*Renewal*

1. Each server  $P_\ell$  selects a random symmetric polynomial

$$r^\ell(x, y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} r_{i,j} x^i y^j,$$

where  $r_{00} = 0$  and  $r_{ij} = r_{ji}$  for all  $i, j$ .

2.  $P_\ell$  sends  $h_k^\ell(x) = r^\ell(x, \omega^k)$  to  $P_k$  for  $k = 1, 2, \dots, n$  by a secure channel, and broadcasts  $h_0^\ell(x) = r^\ell(x, 0)$ .
3.  $P_k$  checks whether  $h_0^\ell(0) = 0$  and  $h_k^\ell(0) = h_0^\ell(\omega^k)$ . If the conditions are satisfied, then  $P_k$  computes and sends to  $P_m$  the value  $h_k^\ell(\omega^m)$ . Otherwise  $P_k$  broadcasts an accusation of  $P_\ell$ .
4.  $P_m$  checks whether  $h_m^\ell(\omega^k) = h_k^\ell(\omega^m)$  for all values of  $\ell$  not accused by  $n - b$  servers of the system. If the equation is not true for more than  $b$  values of  $k$ , then  $P_m$  broadcasts an accusation of  $P_\ell$ .
5. If  $P_\ell$  is accused by at most  $b$  servers, then he can defend himself as follows: For those  $P_i$  he is accused by,  $P_\ell$  broadcasts  $h_i^\ell(x)$ . Then, server  $P_k$  checks whether  $h_i^\ell(\omega^k) = h_k^\ell(\omega^i)$  and broadcasts “yes” or “no”. If there are at least  $n - b - 2$  servers broadcasting yes, then  $P_\ell$  is not a bad server.
6.  $P_m$  updates the list of good servers  $\mathcal{L}$  (i.e., all the values  $\ell$  for which  $P_\ell$  is accused by at least  $b + 1$  servers, or found bad in the previous step are not in  $\mathcal{L}$ ). Then,  $P_m$  updates its shares as

$$h_m(x) \leftarrow h_m(x) + h_m^k(x)$$

for all  $k \in \mathcal{L}$ .

during period  $p$ , he can still use the share  $h_m(0)$  during periods  $p + i$  because he can compute all the updates for this coefficient performed between period  $p$  and period  $p + i$ . More precisely, if the adversary learns the shares  $h_1(0), \dots, h_b(0)$  held by  $S_1, \dots, S_b$  during period  $p$ , and he learns  $h_{b+1}(0), \dots, h_{b+s}(0)$  held by  $S_{b+1}, \dots, S_{b+s}$  during period  $p + 1$  (the adversary is mobile), then, he can compute the new shares held by  $S_1, \dots, S_b$  during period  $p + 1$  from  $h_1(0), \dots, h_b(0)$  and the broadcasts of period  $p + 1$ , and if  $b + s \geq t$  he can recover the secret. Hence, the proactive security property is lost because the renewal scheme does not render useless the shares learnt during the previous period. Exactly the same strategy can be applied to break the Renewal procedure given in [15], which is a generalization of the one given in [21].

Basically, the problem in the above procedure is due to the *broadcast* in Step 2 of  $h_0^\ell(x)$ , needed to verify that the update does not destroy the secret, and the symmetry of  $r^\ell(x, y)$ . We propose two solutions. The first one changes the structure of the renewal phase in order to avoid the broadcast. The second keeps the same structure as before, but removes the symmetry property of  $r^\ell(x, y)$ . Let us describe the first approach: We would like to refresh the shares still by summing up “new shares” derived from a random symmetric polynomial  $r(x, y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} r_{i,j} x^i y^j$  whose known coefficient is  $r_{0,0} = 0$ . Indeed, this



property guarantees that the secret stays the same. However, some server  $P_\ell$  can be bad and can choose a polynomial  $r^\ell(x, y)$  where  $r_{0,0} \neq 0$ . In order to prevent this problem, avoiding the broadcast, we generate  $r(x, y)$  as  $r(x, y) = (x + y)r^*(x, y)$ , where  $r^*(x, y) = \sum_{i=0}^{t-2} \sum_{j=0}^{t-2} r_{i,j}^* x^i y^j$  is given by the sum of the partial choices  $r^\ell(x, y) = \sum_{i=0}^{t-2} \sum_{j=0}^{t-2} r_{i,j}^\ell x^i y^j$  of the good players  $P_\ell$ , and the term  $(x + y)$  is introduced by each server through a private computation. In this way, the condition  $r_{0,0} = 0$  is surely satisfied and the polynomial remains symmetric. From a technical point of view, the degree of  $r(x, y)$  must be  $t - 1$ , in order to enable the reconstruction of the secret. Hence, due to the generation rule for  $r(x, y)$ , every  $r^\ell(x, y)$  must have degree  $t - 2$ .

#### Renewal

1. Each server  $P_\ell$  selects a random symmetric polynomial

$$r^\ell(x, y) = \sum_{i=0}^{t-2} \sum_{j=0}^{t-2} r_{i,j}^\ell x^i y^j,$$

where  $r_{ij} = r_{ji}$  for all  $i, j$ .

2.  $P_\ell$  sends  $h_k^\ell(x) = r^\ell(x, \omega^k)$  to  $P_k$  for  $k = 1, 2, \dots, n$  by a secure channel.
3. After receiving  $h_k^\ell(x)$ ,  $P_k$  computes and sends the value  $h_k^\ell(\omega^m)$  to  $P_m$ , for  $m = 1, \dots, n$ , by a secure channel.
4.  $P_m$  checks whether  $h_m^\ell(\omega^k) = h_k^\ell(\omega^m)$  for  $k = 1, \dots, n$ . If the equation is not true for more than  $b$  values of  $k$ , then  $P_m$  broadcasts an accusation of  $P_\ell$ .
5. If  $P_\ell$  is accused by at most  $b$  servers, then he can defend himself as follows: For those  $P_i$  he is accused by,  $P_\ell$  broadcasts  $h_i^\ell(x)$ . Then, server  $P_k$  checks whether  $h_i^\ell(\omega^k) = h_k^\ell(\omega^i)$  and broadcasts “yes” or “no”. If, for every broadcasted  $h_i^\ell(x)$ , there are at least  $n - b - 2$  servers broadcasting yes, then  $P_\ell$  is not a bad server. In this case, if  $P_i$  has an  $h_i^\ell(x)$  different from the one that  $P_\ell$  has broadcasted, then he stores the broadcasted one.
6.  $P_m$  updates the list  $\mathcal{L}$  of good servers (i.e., the servers found bad in the previous step are not in  $\mathcal{L}$ ) and updates his share as

$$h_m(x) \leftarrow h_m(x) + (x + \omega^m)h_m^*(x)$$

where  $h_m^*(x) = \sum_{\ell \in \mathcal{L}} h_m^\ell(x)$ .

Notice that the above procedure is a slightly revised version of the one we initially proposed [7]: it incorporates the observations and the work done by Nikov et al. [16] on our preprint [7]. See [16] for details.

*Security (Sketch).* The security of the above protocol can be shown by proving that the secret stays the same and the update of the shares cannot be computed by a coalition of bad servers. Concerning the first property, notice that the secret  $s$  is shared by the VSS by means of  $h_1(x), \dots, h_n(x)$ . More precisely, it is the

first coefficient of the polynomial  $h(x, 0)$ . Since during *Renewal* each server  $P_m$  computes a new share as  $h_m(x) \leftarrow h_m(x) + (x + \omega^m)h_m^*(x)$ , implicitly the secret becomes the first coefficient of the new polynomial  $h(x, 0) + xh^*(x, 0)$ , where  $xh^*(x, 0)$  is zero when evaluated at  $x = 0$ . Hence, the secret stays the same.

About the security of the update, notice that if the adversary controls  $b$  servers, say  $S_1, \dots, S_b$ , he can compute at most  $b < t-1$  points  $h_m^*(\omega^1), \dots, h_m^*(\omega^b)$ , which give no information about the polynomial  $h_m^*(x)$  used by  $P_m$  to update his share for any  $m \notin \{1, \dots, b\}$ . Moreover, due to the random choices performed at each executions of *Renewal*, it is not difficult to check that the adversary cannot use the information learnt in period  $p$  during period  $p+1$  or in any other period. Finally notice that, during step 4, a good server  $P_\ell$ , in order to defend himself, broadcasts at most  $b$  polynomials  $h_k^\ell(x)$ , corresponding to the  $P_k$  he is accused by. Assuming that  $t > b+1$ , the polynomials broadcasted give no information about  $r^\ell(x, y)$ . This implies again that an adversary can gain no information about  $h_m^\ell(x)$ , for every  $P_m$  not belonging to the coalition of corrupted servers.

During each time period, the servers need to check if some of them have been corrupted by the adversary. Indeed, those servers should be rebooted<sup>2</sup> in order to recover a correct functionality. The following procedures enable the detection of corrupted servers and the recovering of good shares, once the corrupted servers have been rebooted [21].

#### Detection

1.  $P_\ell$  computes and sends  $h_\ell(\omega^k)$  to  $P_k$  for  $k = 1, 2, \dots, n$  by secure channels.
2.  $P_k$  checks whether  $h_\ell(\omega^k) = h_k(\omega^\ell)$ .  $P_k$  then broadcasts an accusation  $list_k$  which contains those  $\ell$  such that  $h_\ell(\omega^k) \neq h_k(\omega^\ell)$  or  $h_\ell(\omega^k)$  was not received.
3. Each good server updates the list  $\mathcal{L}$  so that it does not contain those  $\ell$  accused by at least  $b+1$  servers of the system.

#### Recovery.

1. For each  $\ell \notin \mathcal{L}$ , every good server  $P_i$  computes and sends  $h_i(\omega^\ell)$  to  $P_\ell$ .
2. Upon receiving the data,  $P_\ell$  computes the polynomial  $h_\ell(x)$  that agree with the majority of the values  $h_\ell(\omega^k)$  he has received.  $P_\ell$  sets  $h_\ell(x)$  as his new share.

<sup>2</sup> We can assume that there is a distributed rebooting scheme enabling a majority of servers to decide to reboot some other servers when they detect that such servers have been corrupted. Otherwise, the system manager who installs the programs, is alerted by the good servers and reboots the bad ones [9].

To understand the above procedures, notice that, when the secret is shared by means of the VSS, the shares held by  $P_i$  and  $P_j$  satisfy  $h_i(\omega^j) = h_j(\omega^i)$ . This property even holds for the polynomials  $h_i^*(x)$  and  $h_j^*(x)$  generated during *Renewal*. Moreover, due to the choice of the updating rule, i.e.,  $h_m(x) \leftarrow h_m(x) + (x + \omega^m)h_m^*(x)$ , the symmetry  $h_i(\omega^j) = h_j(\omega^i)$  is still maintained after every update phase.

These three protocols provide the VSS scheme described in the previous section with proactive security, and they can be used, as we show later, to set up a proactive VDKDS.

The second approach for adding proactive security to the basic VSS given in [21] relies on the use of a generic (non-symmetric) polynomial  $r^\ell(x, y)$ . Let us consider the following procedure:

*Renewal*

1. Each server  $P_\ell$  selects a random polynomial

$$r^\ell(x, y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} r_{i,j} x^i y^j,$$

where  $r_{00} = 0$ .

2.  $P_\ell$  sends  $f_k^\ell(x) = r^\ell(x, \omega^k)$  and  $g_k^\ell(y) = r^\ell(\omega^k, y)$  to  $P_k$  by a secure channel, and broadcasts  $g_0^\ell(x) = r^\ell(x, 0)$ .
3.  $P_k$  checks whether  $g_0^\ell(0) = 0$ ,  $g_0^\ell(\omega^k) = g_k^\ell(0)$ , and  $f_k^\ell(\omega^k) = g_k^\ell(\omega^k)$ . If the conditions are satisfied, then  $P_k$  computes and sends the value  $f_k^\ell(\omega^m)$  to  $P_m$  by a secure channel, for  $m = 1, \dots, n$ . Otherwise,  $P_k$  broadcasts an accusation of  $P_\ell$ .
4.  $P_m$  checks whether  $f_k^\ell(\omega^m) = g_m^\ell(\omega^k)$  for all values of  $\ell$  not accused by  $n - b$  servers of the system. If the equation is not true for more than  $b$  values of  $k$ , then  $P_m$  broadcasts an accusation of  $P_\ell$ .
5. If  $P_\ell$  is accused by at most  $b$  servers, then  $P_\ell$  can defend himself as follows. For those  $P_k$  he is accused by,  $P_\ell$  broadcasts  $f_k^\ell(x)$  and  $g_k^\ell(y)$ . Then, server  $P_i$  checks whether  $g_i^\ell(\omega^k) = f_k^\ell(\omega^i)$ ,  $g_k^\ell(\omega^i) = f_i^\ell(\omega^k)$ , and broadcast “yes” or “no”. If, for every broadcasted pair of polynomials  $(f_k^\ell(x), g_k^\ell(y))$ , there are at least  $n - b - 2$  servers broadcasting yes, then  $P_\ell$  is not a bad server. In this case, if  $P_k$  has a pair  $(f_k^\ell(x), g_k^\ell(y))$  different from the one that  $P_\ell$  has broadcasted, then he stores the broadcasted one.
6.  $P_m$  updates the list of good servers  $\mathcal{L}$  (i.e., the values  $\ell$  for which  $P_\ell$  is accused by at least  $b + 1$  servers, or found bad in the previous step are not in  $\mathcal{L}$ ). Then,  $P_m$  updates his share as

$$h_m(x) \leftarrow h_m(x) + f_m^k(x)$$

for all  $k \in \mathcal{L}$ . Moreover, he updates his information for verification (which is  $g_m(y) = h_m(x)$  at the first execution of *Renewal*) by setting

$$g_m(y) \leftarrow g_m(y) + g_m^k(y)$$

for all  $k \in \mathcal{L}$ . This information is used in the *Detection* procedure.

*Security (Sketch).* The security of the protocol follows from the following observations: first of all notice that from the broadcast  $g_0^\ell(x) = r^\ell(x, 0) \neq r^\ell(0, y)$ , the value  $f_k^\ell(0) = r^\ell(0, \omega^k)$  cannot be computed. Moreover, every participant, during steps 3 and 4, checks that the update does not destroy the shared secret, and that the polynomials they have received are consistent. Moreover, as we have already seen before, the condition  $t > b + 1$  ensures that the polynomials broadcasted in step 5 by  $P_\ell$ , to defend himself against at most  $b$  bad  $P_i$ , do not give any information about  $r^\ell(x, y)$ , and hence do not give any information about  $f_m^\ell(x)$  for any  $P_m$  not belonging to the coalition of bad servers.

This procedure can be applied to both VSSs given in [21] and [8] when  $b \leq \frac{n}{4} - 1$ . In fact, when applied to the scheme in [8], the polynomial  $g_m(y)$  in Step 5 at the first execution of *Renewal* is already different from  $h_m(x)$ : it is the polynomial  $g_m(y)$  used for verification given by the VSS described in [8]. Actually, the above procedure has the structure of the procedure given in [21], but it has been modified according to the design of the VSS given in [8].

The following protocols enable the detection of corrupted servers and the recovering of good shares for the rebooted servers.

*Detection.*

1.  $P_\ell$  computes and sends  $h_\ell(\omega^k)$  to  $P_k$  for  $k = 1, 2, \dots, n$  by secure channels.
2.  $P_k$  checks whether  $h_\ell(\omega^k) = g_k(\omega^\ell)$ .  $P_k$  then broadcasts an accusation  $list_k$  which contains those  $\ell$  such that  $h_\ell(\omega^k) \neq g_k(\omega^\ell)$  or  $h_\ell(\omega^k)$  was not received.
3. Each good server updates the list  $\mathcal{L}$  so that it does not contain those  $\ell$  accused by at least  $b + 1$  servers of the system.

*Recovering*

1. For each  $\ell \notin \mathcal{L}$ , every good server  $P_i$  computes and sends  $h_i(\omega^\ell)$  and  $g_i(\omega^\ell)$  to  $P_\ell$ .
2. Upon receiving the data,  $P_\ell$  computes two polynomials  $h_\ell(x)$  and  $g_\ell(y)$  that agree with the majority of the values  $h_\ell(\omega^k)$  and  $g_\ell(\omega^k)$  it has received.  $P_\ell$  sets  $h_\ell(x)$  as its share and  $g_\ell(y)$  as its verification information.

We would like to point out that both the *Renewal* phases described before can be implemented by using *random one-time pads* and the *broadcast channel*, instead of using secure channels for the checks of consistency of the shares. Such an approach enables saving one round of communication, but the resulting procedures are perhaps less readable than the previous ones.

## 5.2 A Proactive VDKDS

At this point, we have all the tools to set up a Proactive VDKDS. To summarize:

- In the protocol for a VDKDS, described in Section 4, the keys are values of a polynomial whose coefficients are (verifiably) shared among the servers. More precisely, to set up the DKDC, each server  $P_m$  chooses a *random* polynomial

$$K^m(x) = \sum_{z=0}^{\ell-1} k_z^{(m)} x^z.$$

Then,  $P_m$  uses  $\ell$  different instances of the VSS given in Section 3, i.e., one for each coefficient, to distribute in a verifiable way the coefficients of his polynomial  $K^m(x)$ . According to the VSS, each server  $P_k$  receives  $\ell$  polynomials from  $P_m$ , one for each coefficient  $k_z^{(m)}$ . The conference key for  $C_s$  is then defined to be  $\kappa_s = K(s)$ , where

$$K(x) = \sum_{z=0}^{\ell-1} k_z x^z = \sum_{m \in \mathcal{L}} K^m(x)$$

and  $\mathcal{L}$  is the list of good servers. At the end of the set up phase, every server  $P_k$  stores  $\ell$  polynomials,  $h_k^{k_0}(x), \dots, h_k^{k_{\ell-1}}(x)$ , each sharing one coefficient of  $K(x)$ , by summing up the partial shares/polynomials received for each coefficient  $k_z^{(m)}$  from servers  $P_m$  belonging to the list of good servers.

- Therefore, a straightforward solution to gain proactive security could be to directly apply, at the beginning of each time period, the *Detection*, *Recovery* and *Renewal* procedures for each coefficient of the polynomial  $K(x)$ , generated by the good servers during the set up phase of the system.

## 6 Conclusions

In this paper we have shown how to set up a Robust Distributed Key Distribution Scheme, enabling a set of servers to jointly realize a Key Distribution Center. We have used unconditionally secure verifiable proactive secret sharing schemes as a building block. As well, we have revised the unconditionally secure VSS described by Stinson and Wei in [21], proposing a modified version which is proactively secure. Moreover, we have given proactive routines that can be applied to both schemes given in [21, 8] when  $b < \frac{n}{4}$ . Since the proactive security property can be useful in several settings in which the adversary is mobile, the applicability of such schemes has independent interest of the specific application to key distribution that has been addressed in this paper. In the full version of the paper we will provide complete proofs, and the case in which the number of bad servers is  $b < \frac{n}{3}$  will be considered as well.

## Acknowledgements

We would like to thank Svetla Nikova, Ventzislav Nikov and Ruizhong Wei for helpful comments and discussions during the writing of this paper. D. R. Stinson's research is supported by NSERC grants IRC # 216431-96 and RGPIN # 203114-02.

## References

1. M. Bellare and P. Rogaway, *Provably Secure Session Key Distribution: The Three Party Case*, Proc. of the 27th Annual Symposium on the Theory of Computing (STOC '95), ACM, pp. 57–66, 1995.
2. C. Blundo, and P. D'Arco, *Unconditionally Secure Distributed Key Distribution Schemes*, submitted for publication.
3. C. Blundo and P. D'Arco, *The Key Establishment Problem*, Lecture Notes in Computer Science, FOSAD 2001 (Tutorial), to appear.
4. C. Blundo, P. D'Arco, V. Daza and C. Padrò. *Bounds and Constructions for Unconditionally Secure Distributed Key Distribution Schemes for General Access Structures*, Proc. of the Information Security Conference (ISC 2001), Lecture Notes in Computer Science, vol. 2200, pp. 1–17, 2001.
5. B. Chor, S. Goldwasser, S. Micali, and B. Awerbach. *Verifiable Secret Sharing and Achieving Simultaneity in Presence of Faults*, Proc. of the 26-th Annual Symposium on the Foundations of Computer Science, IEEE, pp. 383–395, 1985.
6. P. D'Arco, *On the Distribution of a Key Distribution Center* (extended abstract), Proc. of the Italian Conference on Theoretical Computer Science (ICTCS '01), Lecture Notes in Computer Science, vol. 2202, pp. 357–369, 2001.
7. P. D'Arco and D. R. Stinson, *On Unconditionally Secure Proactive Verifiable Secret Sharing Schemes and Distributed Key Distribution Centers*, unpublished manuscript, May 2002.
8. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin, *The Round Complexity of Verifiable Secret Sharing and Secure Multicast*, Proc. of the 33-rd Annual Symposium on the Theory of Computing (STOC '01), ACM, pp. 580–589, 2001.
9. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. *Proactive Secret Sharing or: How to Cope with Perpetual Leakage*, Advances in Cryptology - Crypto '95, Lecture Notes in Computer Science, vol. 963, pp. 339–352, 1995.
10. F. J. MacWilliams and N. J. A. Sloane, **The Theory of Error-Correcting Codes**, North-Holland, Amsterdam, 1981.
11. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, **Handbook of Applied Cryptography**, CRC Press, 1996.
12. M. Naor, B. Pinkas, and O. Reingold. *Distributed Pseudo-random Functions and KDCs*, Advances in Cryptology - Eurocrypt'99, Lecture Notes in Computer Science, vol. 1592, pp. 327–346, 1999.
13. R. M. Needham and M. D. Schroeder. *Using Encryption for Authentication in Large Networks of Computers*, Communications of ACM, vol. 21, pp. 993–999, 1978.
14. B. C. Neuman and T. Tso. *Kerberos: An Authentication Service for Computer Networks*, IEEE Transactions on Communications, vol. 32, pp. 33–38, 1994.
15. V. Nikov, S. Nikova, B. Preneel and J. Vandewalle. *Applying General Access Structure to Proactive Secret Sharing Schemes*. Proc. of the 23rd Symposium on Information Theory in the Benelux, May 29-31, 2002, Université Catholique de Louvain (UCL), Louvain-la-Neuve, Belgium.
16. V. Nikov, S. Nikova, B. Preneel and J. Vandewalle. *On Distributed Key Distribution Centers and Unconditionally Secure Proactive Verifiable Secret Sharing Schemes Based on General Access Structures*, preprint, August 2002.
17. R. Ostrovsky and M. Yung, *How to Withstand Mobile Virus Attacks*, Symposium on Principles of Distributed Computing (PODC '91), ACM, pp. 51–59, 1991.

18. R. S. Rees, D. R. Stinson, R. Wei, and G. H. J. van Rees, *An Application of Covering Designs: Determining the Maximum Consistent Set of Shares in a Threshold Scheme*, *Ars Combinatoria* **53**, 225–237, 1999.
19. D.R. Stinson, **Cryptography: Theory and Practice**, CRC Press, 1995 (2nd Edition, 2002).
20. D. R. Stinson. *On Some Methods for Unconditional Secure Key Distribution and Broadcast Encryption*, *Designs, Codes and Cryptography*, vol. 12, pp. 215–243, 1997.
21. D. Stinson and R. Wei, *Unconditionally Secure Proactive Secret Sharing Scheme with Combinatorial Structures*, *SAC'99. Lecture Notes in Computer Science*, vol. 1758, pp. 200–214, 1999.

## A A $(k, n, \mathcal{C}, \mathcal{G})$ -DKDS

In a  $(k, n, \mathcal{C}, \mathcal{G})$ -DKDS, each user can compute a common key by interacting with any  $k$ -subset of the  $n$  servers at his choice. In [12], a construction based on bivariate polynomials for a  $(k, n, \mathcal{C}, \mathcal{G})$ -DKDS was proposed. Basically, it works as follows: Each of the servers  $S_1, \dots, S_k$ , performing the initialization phase, constructs a random bivariate polynomial  $P^i(x, y)$  of degree  $k-1$  in  $x$ , and  $\ell-1$  in  $y$ , and sends  $Q_j^i(y) = P^i(j, y)$  to the server  $S_j$ , for  $j = 1, \dots, n$ . Server  $S_j$  computes his private information,  $Q_j(y)$ , by adding the  $k$  polynomials received from  $S_1, \dots, S_k$ . A user who wants to compute a conference key,  $\kappa_h$ , sends to (at least)  $k$  servers a key request. Each server  $S_j$ , invoked by the user, checks that the user belongs to  $C_h$ , and sends to the user the value  $Q_j(h)$ . Using the  $k$  values received from the servers, and applying the Lagrange formula for polynomial interpolation, each user in  $C_h$  recovers the secret key  $P(0, h) = \sum_{i=1}^k P^i(0, h)$  (see [12] for details).

The construction is correct and secure, according to the model considered in [12]. In order to introduce verifiability and proactivity, the following approach was suggested in [12]. Time is divided in periods. At the beginning of period  $t$ , for  $i = 1, \dots, k$ , each server  $S_i$  performing the initialization, chooses a random polynomial  $P_i^t(x, y)$  of degree  $k-1$  in  $x$  and  $\ell-1$  in  $y$  such that  $P_i^t(0, h) = 0$  for each  $h \in Z_q$ . Then, for  $i = 1, \dots, k$ , server  $S_i$  sends, for  $j = 1, \dots, n$ , the univariate polynomial  $Q_{i,j}^t(y) = P_i^t(j, y)$  to server  $S_j$ , and broadcasts the univariate polynomial  $P_i^t(x, c)$ , where  $c$  is a public point. Then, for  $j = 1, \dots, n$ , server  $S_j$  checks that  $P_i^t(x, c)$  evaluated in  $x = 0$  is zero (i.e.,  $P_i^t(0, c) = 0$ ) and that the broadcasted polynomial is consistent with  $Q_{i,j}^t(y)$  (i.e.,  $Q_{i,j}^t(c) = P_i^t(j, c)$ ). Finally, if the check is satisfied,  $S_j$  updates his private information by computing  $Q_j(y) \leftarrow Q_j(y) + \sum_{i=1}^k Q_{i,j}^t(y)$ . Unfortunately, a server sending information during the update phase can cheat, as shown by the following example.

**Example.** Let us consider a  $(3, 3, \mathcal{C}, \mathcal{G})$ -DKDS. The polynomial  $P_i^t(x, y)$  chosen by  $S_i$  at the beginning of the period in order to update the system is of degree 2 in  $x$  and  $\ell-1$  in  $y$ . A cheating  $S_i$  can choose  $P_i^t(x, y) = a + b_1x + b_2x^2 + P_Y(y)$  where  $a = -P_Y(c)$  and  $P_Y(y) = \sum_{j=1}^{\ell-1} p_j y^j$ . It is not difficult to check that  $P_i^t(0, c) = 0$  and that  $P_i^t(x, c) = b_1x + b_2x^2$  is equal to  $Q_{i,j}^t(y)$  when the first one is evaluated in  $j$  and the second one in  $c$ . But  $P_i^t(0, c') \neq 0$  for any  $c' \neq c$ .

# Short Signatures in the Random Oracle Model

Louis Granboulan\*

École Normale Supérieure,  
Louis.Granboulan@ens.fr

**Abstract.** We study how digital signature schemes can generate signatures as short as possible, in particular in the case where partial message recovery is allowed. We give a concrete proposition named OPSSR that achieves the lower bound for message expansion, and give an exact security proof of the scheme in the ideal cipher model. We extend it to the multi-key setting. We also show that this padding can be used for an asymmetric encryption scheme with minimal message expansion.

**Keywords:** digital signature, padding, random oracle and ideal cipher models, proven security.

## 1 Introduction

### 1.1 Overview of the Results

A digital signature scheme allows a signer to transform an arbitrary message into a signed message, such that anyone can check the validity of the signed message using the signer's public key, but only the signer is able to generate signed messages. A signed message contains the information about the message, plus some information to prove its validity. For example in the case of a scheme without message recovery, the signed message is the concatenation of the message and of a signature.

The message expansion of a signature scheme is the difference between the length of the signed message and the original message. It is the length of the signature, if there is no message recovery. We show how to obtain message expansion as small as possible, with a concrete scheme having proven security in the ideal cipher model. The OPSSR technique is a padding for schemes based on trapdoor one-way bijections. Its performance cost is small, and its security is similar to the other schemes in the hash-then-invert paradigm.

The paper is organized as follows. Section 2 describes a formalism for digital signature schemes and describes the properties of the RSA trapdoor one-way bijection. Section 3 shows what are the lower bounds for message expansion. Section 4 describes OPSSR, which has minimal message expansion. Section 5 raises and solves a theoretical problem that arises when having an idealized security

---

\* Part of this work has been supported by the Commission of the European Communities through the IST Programme under Contract IST-1999-12324 (NESSIE). This paper is NESSIE document **NES/DOC/ENS/WP5/021/2**.



model for a multi-key setting. Section 6 discusses open problems. Appendix A compares OPSSR with other paddings. Appendix B explains why OPSSR can also be used for encryption.

## 1.2 Related Work

Many schemes have been proposed with short signatures [4,11,12,17,18], but their exact security is not proven to be equivalent to the underlying problem with the same parameters, because their security proofs are not tight. Therefore if the parameters are chosen to give short signatures, the security of those schemes is not proven.

Partial message recovery can allow one to reduce message expansion when having security parameters corresponding to the tightness of the security proof. Message recovery has been used to reduce the message expansion in the PSS scheme [2], the Pintsov-Vanstone scheme [5] or the DSA-like schemes [17]. But those schemes do not achieve minimal signature length.

Coron [9] has shown how to reduce the length of the random salt in PSS, to improve the amount of message recovered, and reduce the message expansion. But the result of this improvement is still not optimal.

Coron, Joye, Naccache and Paillier [10] have shown that the PSS padding, which was designed for signature, can also be used for encryption.

## 1.3 Our Contribution

We introduce the definition of message expansion which generalizes the notion of short signature for schemes with message recovery. We show what is the minimal possible message expansion for a given proven security requirement. We describe a padding that achieves this lower bound and that can be used with RSA. This padding can be viewed as a generalization of PSSR and many other paddings. We also show that most current schemes proven secure in a idealized model should go under a small modification that increases their security in the multi-key setting.

# 2 Definitions

## 2.1 Digital Signature Schemes

**Notations.** If the variable  $x$  represents a value taken from a finite set  $\mathcal{X}$  of  $n$  elements, then we say that the size of  $x$  is the value  $\#x = \#\mathcal{X} = \log_2 n$ , which may not be an integer.

If the elements of  $\mathcal{X}$  can be represented by bit strings, then  $\sharp x = \sharp \mathcal{X}$  is the length of the bit strings. Of course,  $\#x \leq \sharp x$ .

For variables  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ , the corresponding element of  $\mathcal{X} \times \mathcal{Y}$  is written as  $x\|y$ . This notation comes from the fact that if  $x$  and  $y$  are bit strings, then  $x\|y$  is the concatenation of these strings.

**Definitions.** A signature scheme is described by the following four algorithms:

- a parameter generation  $\text{Generate} : \rho \mapsto \text{param}$ ,
- a key generation  $\text{KeyGen}_{\text{param}} : \rho' \mapsto (\text{pk}, \text{sk})$ ,
- a signature generation  $\text{Sign}_{\text{param}, \text{pk}, \text{sk}} : (m, r) \mapsto \sigma$
- and a signature verification  $\text{Ver}_{\text{param}, \text{pk}} : (\sigma, r') \mapsto m \text{ or reject}$ .

All these algorithms are deterministic, and the inputs  $\rho$ ,  $\rho'$ ,  $r$  and  $r'$  (if non empty) contain the randomization for the algorithms. They may have some specific format.

Signature schemes with appendix have the property that  $\sigma = m \| s$ . Signature schemes with message recovery usually have the property that  $\sigma = \hat{m} \| s$  and that the whole message is  $m = \hat{m} \| \bar{m}$ , where  $\bar{m}$  is the recovered part of the message. They typically have a lower bound for the size of the whole message, which is also the amount of message recovered.<sup>1</sup> Signature schemes with unique signature have the property that  $\text{Ver}$  is injective (with the exception of `reject`), which also implies that  $\text{Sign}$  does not use any random  $r$  (deterministic signature scheme).

Two signature schemes are equivalent when the following conditions are satisfied:

- The possible values of  $\text{param}$  are the same.
- The distributions of the  $\text{pk}$  generated are indistinguishable.
- Both verification algorithms are the same.
- The output of the respective  $\text{Sign}$  operations for fixed  $m$  and random  $r$  are indistinguishable.

## 2.2 Security Model and Proofs

A  $(t, \varepsilon, q_S)$ -forger is able to make  $q_S$  queries for signatures and tries to produce a new valid signature. It succeeds in time  $t$  with probability  $\varepsilon$ . A signature scheme with no  $(t, \varepsilon, q_S)$ -forger is said to be  $(t, \varepsilon, q_S)$ -secure against adaptive chosen message attack. This security also means non-repudiation, because it proves that only the signer is able to make valid signed messages.

Weak security means that the forgery should be a valid signed message for a message that was not the input of a query. Strong security means that the forgery should be a valid signed message that was not the answer of a query. These notions are equivalent if the scheme has unique signature.

The security level of a scheme is  $k$  bits if there exists no  $(t, \varepsilon, q_S)$ -forger with  $\log_2(t/\varepsilon) < k$ . This value  $k$  depends of the time unit used for  $t$ .

Please note that any  $(t, \varepsilon, q_S)$ -forger for a signature scheme is also a  $(t, \varepsilon, q_S)$ -forger for all equivalent signature schemes.

A mathematical problem is  $(t', \varepsilon')$ -secure if there exist no algorithm that solves an arbitrary instance of the problem in time  $t'$  with probability better than  $\varepsilon'$ . The difficulty is  $k'$  bits if there exist no  $(t', \varepsilon')$ -solver with  $\log_2(t'/\varepsilon') < k'$ .

---

<sup>1</sup> This lower bound can be overcome by storing the length of the actual recovered part in  $\bar{m}$ . E.g. by padding  $\bar{m}$  with a 1 followed by a string of 0. With this padding, one bit of message expansion is added.

A proof of security is the description of how to construct a  $(t', \varepsilon')$ -solver (called *reduction algorithm*) when given access to a  $(t, \varepsilon, q_S)$ -forger. The reduction simulates an equivalent signature scheme and answers signature queries from the forger. The forgery is used to solve the problem. The reduction does not always succeed, partly because its simulation of an equivalent signature scheme may not be perfect, and partly because the forgery may be useless.

A tight proof of security has  $t'/\varepsilon' \simeq t/\varepsilon$ .

## 2.3 Idealized Models

An idealized oracle model replaces some components of the verification algorithm with calls to an oracle which is simulated by the reduction. The number of calls to these oracles is bounded e.g. by  $q_O$ . Because the actual computation of the idealized components takes time, a scheme with  $k$  bits of security with the appropriate time unit always has  $q_O \leq 2^k$ .

The random oracle model replaces hash functions by calls giving random output. The generic group model replaces the operations in some group by random answers that respect the group laws. The random permutation model replaces a fixed permutation by a random one constructed in answer to the oracle calls. The ideal cipher model replaces a keyed permutation by a random one constructed in answer to the oracle calls.

A reduction algorithm in a idealized model always gives random answers taken from the set of values that are consistent with previous answers. It has a total freedom for its answer to the first oracle query, and the other answers should not allow the forger to detect that the reduction algorithm took control of the oracle. Consistency for a random oracle means that the same input always give the same output. For a random permutation, two different inputs have different outputs, and queries for the inverse permutation should also be consistent.

To be able to maintain consistency, the reduction algorithm needs to keep tables of the subset of input/output pairs that has been developed to answer the queries. In other words, the reduction algorithm constructs the oracle tables.

The random oracle model is widely used in the literature, the ideal cipher model and the generic group model have been used for proving the security of some specific schemes. Proofs in these models cannot generically be translated into the real world [6,13], but it is widely believed that a proof in an idealized model give some confidence in the design of a cryptographic primitive. The random oracle model and ideal cipher model are very similar and we believe that they give similar confidence in cryptographic designs: a random oracle can be constructed from ideal ciphers, and it might be possible to build an ideal cipher from random oracles.

## 2.4 The RSA Trapdoor One-Way Bijection

**Bijection.** A bijection with length  $l$  is a one-to-one and onto mapping  $F$  from a set  $\mathcal{S}$  with  $2^l$  elements to a set  $\mathcal{L}$  with  $2^l$  elements. It is a permutation if  $\mathcal{S} = \mathcal{L}$ . Let  $l'$  be equal to  $\#S$ .

**One-Way.** A bijection with length  $l$  is one-way with security  $k'$  bits if  $F$  is easy to compute but finding the preimage for a random  $y \in \mathcal{L}$  (i.e. the unique  $x \in \mathcal{S}$  such that  $y = F(x)$ ) is a problem with a difficulty of  $k'$  bits. Exhaustive search in  $\mathcal{S}$  shows that  $k' \leq l'$ .

**Trapdoor.** It is a trapdoor one-way bijection if knowing some secret information (the trapdoor) makes easy the computation of  $F^{-1}$ .

**Random-Self-Reducibility.** The permutation is random-self-reducible if it has the following additional property. There exists a probabilistic algorithm  $R$  that takes an input  $y \in \mathcal{L}$  and generates a uniformly distributed value  $\tilde{y} \in \mathcal{L}$  such that knowing the value of  $F^{-1}(\tilde{y})$  makes it easy to compute  $F^{-1}(y)$ .

If  $F$  is random-self-reducible, then it is always possible to compute  $F^{-1}(y)$  in time  $2^{l/2}$ , using the birthday paradox. A table of  $2^{l/2}$  random  $(x, F(x))$  pairs is computed. A table of  $2^{l/2}$  random  $\tilde{y}$  values is generated with  $R(y)$ . A collision  $F(x) = \tilde{y}$  gives the value for  $F^{-1}(\tilde{y})$ , from which we deduce the value of  $F^{-1}(y)$ . For a random-self-reducible trapdoor one-way permutation, we always have  $k' \leq l/2$ .

**RSA Permutation.** The public parameter is a number  $n$  and an odd exponent  $e$ , the corresponding secret is the factorization  $pq = n$  or the inverse  $e^{-1} \bmod \phi(n)$ . The function  $F(x) = x^e \bmod n$  is a permutation of the set  $\mathbb{Z}_n^*$  of invertible integers modulo  $n$ . The trapdoor owner can compute  $F^{-1}(x) = x^{e^{-1}} \bmod n$ .

This function  $F$  is a random-self-reducible trapdoor one-way permutation. Its random-self-reducibility comes from the algorithm  $R$  that generates a random  $\tilde{x} \in \mathbb{Z}_n^*$  and returns  $\tilde{y} = y \cdot \tilde{x}^e$ . Then  $F^{-1}(y) = F^{-1}(\tilde{y})/\tilde{x}$ .

The best known technique to compute  $F^{-1}$  is to compute the factorization of  $n$ . Here is a table that gives estimates for minimal bit length of  $n$  to have some given security levels. The problem of the estimation of the difficulty of factoring large numbers is the object of some controversies and this table should only be understood as a proposal for basing our numbers on realistic estimates. It is not an attempt to solve this controversy. It is based on the hypothesis that the recent factorizations of 512 bits numbers needed a workfactor of  $2^{56}$  and that the asymptotic complexity of the number field sieve is around  $L_n[\frac{1}{3}, 1.9]$ .

The formula for the following table is  $k' = 12 + \log(L_{2^l}[\frac{1}{3}, 1.9])$ .

Modulus length  $l$     512 768 1536 4096 8192

Bit security     $k'$     56   64   80   128   160

**RSA Bijection.** For the RSA permutation the permuted set  $\mathcal{L}$  is  $\mathbb{Z}_n^*$  therefore the length  $l$  is not an integer. If an integer value is preferred, the RSA bijection is defined as follows.

The set  $\mathcal{L}$  contains all integers in  $\mathbb{Z}_n$  smaller than  $2^l$ , and  $\mathcal{S}$  is its preimage and  $l' = \lceil l \rceil$ . The computation of  $F(x)$  for  $x \in \mathbb{Z}_n$  begins with  $y = x^e \bmod n$ . If  $y \in \mathcal{L}$ , it is the answer, else  $x$  is rejected because it is not an element of  $\mathcal{S}$ .

### 3 Minimal Message Expansion

#### 3.1 The Lower Bound

A simple counting argument shows that for any signature scheme with random salt of length  $\#r$  and message expansion  $\lambda$ , a signed message is valid with probability at least  $1/2^{\lambda-\#r}$ . Therefore the security level of the scheme is at most  $\lambda - \#r$ .

**Theorem 1.** *Minimal message expansion for  $k$  bits of security is  $k$  bits of message expansion and can only be obtained for a signature scheme with unique signature.*

None of the previously published techniques achieve this lower bound: they don't allow one to go under  $2k$  bits of message expansion. Our OPSSR scheme achieves this lower bound.

#### 3.2 Signature Schemes with Appendix

Coron [9] proved that a signature scheme with unique signature cannot have a tight security proof, and that the lower bound for the relation between the security  $k$  of the scheme and the security  $k'$  of the underlying problem is  $k' \simeq k + \log_2 q_S$ .

A signature scheme with appendix based on a problem with security  $k'$  has an appendix of length at least  $k'$ . Therefore the message expansion for a deterministic signature scheme with appendix is at least  $k + \log_2 q_S$ .

Randomized signature schemes can enhance the tightness of the proof, but at the cost of a random seed that appears in the signed message. Each bit of gained tightness costs one bit of random seed.

**Theorem 2.** *The lower bound for a signature scheme with appendix having  $k$  bits of security against a forger allowed to make  $q_S$  signature queries is a message expansion of  $k + \log_2 q_S$  bits.*

None of the previously published techniques achieves this lower bound, and the problem is still open whether it is possible to achieve it or not.

### 4 The OPSSR Padding

#### 4.1 Some Previous Work: PFDH and PSSR

**Quick Introduction.** Full Domain Hash was formally described and proved by Bellare and Rogaway in [2]. Their proof shows that in the random oracle model with at most  $q_H$  hash queries the security  $k$  of FDH is related to the security  $k'$  of the underlying trapdoor one-way bijection by  $k' \simeq k + \log_2(q_H + q_S)$ . Coron has shown in [8] that random-self-reducibility helps to improve the proof and obtains  $k' \simeq k + \log_2 q_S$ . Coron also introduced in [9] a probabilistic variant of Full Domain Hash that we describe below.

**PFDH.** The two components are a random-self-reducible trapdoor one-way bijection  $F$  and a cryptographic hash function  $H$ . The verification of a signed message splits  $\sigma = m\|r\|s$  and says the signature is valid if  $s \in \mathcal{S}$  and  $H(m\|r) = F(s)$ . It outputs the message  $m$ .

The trapdoor owner signs the message  $m$  by first generating a random salt  $r$ , then computing  $s = F^{-1} \circ H(m\|r)$ , and returns  $\sigma = m\|r\|s$ .

The proof shows that if  $\#r \geq \log_2 q_S$  then  $k' \simeq k$  and if  $\#r \leq \log_2 q_S$  then  $k' \simeq k + \log_2 q_S - \#r$ . We can notice that the output length of the hash function is equal to the length  $l$  of the bijection and that the message expansion is  $l' + \#r$ . Because of random-self-reducibility,  $l' \geq 2k'$ . PFDH does not allow better message expansion than  $2k$ .

**PSSR.** This scheme was introduced in [2] and its optimal proof of security is in [9]. It is a modification of PFDH by adding recovery of the salt and of part of the message.

The hash function  $H$  has output length  $2k$  and an additional cryptographic hash function  $G$  with input length  $2k$  and output length  $l - 2k$  is needed and is modeled as a random oracle.

The verification splits  $\sigma = \hat{m}\|s$ , checks that  $s \in \mathcal{S}$ , computes  $a\|h = F(s)$  and  $\bar{m} = a \oplus G(h)$ , and checks if  $H(\hat{m}\|\bar{m}) \neq h$ . It computes  $m\|r = \hat{m}\|\bar{m}$  and outputs the message  $m$ .

The trapdoor owner signs the message  $m$  by first generating a random salt  $r$ , then computing  $\hat{m}\|\bar{m} = m\|r$  where  $\bar{m}$  is  $l - 2k$  bits long. Then  $h = H(m\|r)$  and  $a = \bar{m} \oplus G(h)$  are computed. The signed message is  $\hat{m}\|F^{-1}(a\|h)$ . PSS is the special case where  $\#r = l - 2k$ .

The security proof is very similar to the proof for PFDH and shows that PSSR has the same security as PFDH. The addition of  $G$  does not weaken the scheme because the probability of a collision in the input of  $G$  is low. This is due to the fact that the input size of  $G$  is twice the security level of the scheme. The message expansion with PSSR is  $2k + \#r$ . PSSR does not allow better message expansion than  $2k$ .

**Replacing a XOR with a Block Cipher.** The idea of improving a padding by replacing a XOR with a block cipher was introduced by Jonsson [14] for an improvement of OAEP+ named OAEP<sup>++</sup>. The same can be done with PSS. It only changes the security properties of the padding when used for asymmetric encryption.

## 4.2 Basic OPSSR

OPSSR means Optimal Padding for Signature Schemes with message Recovery. We begin with a simplified version of our OPSSR scheme.

This signature scheme can only sign messages of length  $l - k$ . It has two parameters: a trapdoor one-way bijection  $F$  with length  $l$  and security  $k'$  and an arbitrary permutation  $E$  of blocks of size  $l$ . The random permutation model for

$E$  is used. In practice  $E$  can be based on a large block cipher with fixed key 0 and  $F$  can be the RSA bijection.

Let  $\kappa$  be a fixed value of  $k$  bits, e.g.  $0^k$ . Valid signatures are generated by  $m \mapsto F^{-1}(E^{-1}(m\|\kappa))$ . The verification computes  $m\|v = E(F(\sigma))$  and checks if  $v \stackrel{?}{=} \kappa$ .

**Security Proof.** We show how it is possible to compute  $F^{-1}(y)$  for an arbitrary  $y$  without knowing the trapdoor, but with access to a forger of OPSSR in the random permutation model.

The number of signature queries is bounded by  $q_S$  and the number of oracle queries (to  $E$  and  $E^{-1}$ ) is bounded by  $q_O$ . For all answers to the  $q_O + q_S \leq 2^k$  queries made by the forger, we will need to generate a value  $y'$  uniformly distributed in  $\mathcal{L}$ . In this proof, one query has  $y' = y$  and all other queries have  $y' = F(x')$  for a random  $x'$ . A table of  $(y', x')$  is stored, enabling the lookup of  $F^{-1}(y')$ .

First we send to the forger the description of  $F$ . Then we will answer to four types of queries and the oracle table is updated according to these answers.

- In response to a signature query for  $m$ , the reduction generates a value  $y'$  and updates the oracle table with  $y' \stackrel{E}{\mapsto} m\|\kappa$ . The answer is  $x' = F^{-1}(y')$ .

The signature query aborts if  $E(y')$  was already defined. Since  $y'$  is uniformly distributed in  $\mathcal{L}$ , and at most  $2^k$  values were defined, this happens with probability at most  $1/2^{l-k}$ .

The signature query also aborts if  $y' = y$ . This has probability  $1/2^k$ .

- In response to a query for  $E^{-1}(m\|\kappa)$ , that is not in the table, a signature query for  $m$  is simulated. The answer is  $y'$ .

The oracle query aborts if  $E(y')$  was already defined. This has probability at most  $1/2^{l-k}$ .

The oracle query does not abort if  $y' = y$ . If the forger later makes a query of a signature for  $m$ , then the signature query will abort.

- In response to a query for  $E^{-1}(m\|v)$  with  $v \neq \kappa$ , a random value  $y''$  is generated and the oracle table is updated with  $y'' \stackrel{E}{\mapsto} m\|v$ .

The oracle query aborts if  $E(y'')$  was already defined. This has probability at most  $1/2^{l-k}$ .

- In response to a query for  $E(y'')$ , random  $m$  and  $v$  are chosen, and the oracle table is updated with  $y'' \stackrel{E}{\mapsto} m\|v$ .

The oracle query aborts if  $v = \kappa$ . This has probability  $1/2^k$ .

If  $l \geq 2k + 1$ , then no query make the reduction abort with probability more than  $2^{-k}$ . The total probability of non abortion is  $(1 - 1/2^k)^{2^k} \geq 1/e$ .

The forger returns a forgery  $\sigma$  which is the signature of a message  $m$  with probability better than  $1/2^k$ . If this message was not in a query for  $E^{-1}(m\|\kappa)$ , then the signature is valid with probability  $1/2^k$ . Therefore this message was in a query for  $E^{-1}(m\|\kappa)$  and a value  $y'$  was generated. The reduction can compute  $F^{-1}(y)$  if this forgery corresponds to  $y = y'$ , which happens with probability  $2^{-k}$ . Therefore the success probability of the reduction is the one of the forger divided by at most  $e2^k$ .

The running time  $t$  of the (real world) forger includes some actual computations of  $F$ ,  $E$  and  $E^{-1}$ . The answer to an oracle query by the reduction algorithm needs some table lookups and at most one computation of  $F$ . Under the hypothesis that the time for all these computations are similar, the running time for the reduction is  $\gamma t$  for some small constant  $\gamma$ .

A difficulty level of  $k' \simeq 2k$  is needed and this scheme has minimal message expansion.

**Random-Self-Reducibility.** The same technique as in [8] can be used when  $F$  is random-self-reducible. This technique consists in a change of the way the values  $y'$  are generated. The full details on how to optimize the parameters can be found in Coron's papers.

The basic idea is to have a proportion  $\alpha/q_S$  of the values  $y'$  generated with the algorithm  $R$ . A signature query will abort if such a  $y'$  was generated, which happens with probability  $\alpha$ . However, if the reduction does not abort, then its success probability is the success probability of the forger divided by  $q_S/\alpha$ .

This idea applies to OPSSR as well and a difficulty level of  $k' \simeq k + \log_2 q_S$  is needed and the scheme has minimal message expansion.

**Randomization.** The same technique as in [29] can be used to enhance the tightness of the reduction, if  $F$  is random-self-reducible. The message  $m$  is padded with a random salt  $r$  before being signed. The signature verification works as before but the salt is discarded.

The reason why this improves the tightness of the reduction is that a much higher proportion of the values  $y'$  can be generated with the algorithm  $R$ , because a signature query can choose a value for the salt for which  $y' = F(x')$ .

This idea applies to OPSSR as well and a difficulty level of  $k' \simeq k + \log_2 q_S - \#r$  is sufficient when the salt has length  $\#r \leq \log_2 q_S$ . However this randomized scheme does not have minimal message expansion, because the salt is recovered and the expansion is  $k + \#r$ .

### 4.3 OPSSR

Basic OPSSR only allows one to sign messages of length  $l - k$ . To sign a message  $m$  of arbitrary length greater than  $l - k$ , the message is split  $\hat{m} \parallel \bar{m} = m$  where  $\bar{m}$  has length  $l - k$  bits.  $\hat{m}$  will be transmitted in the clear and  $\bar{m}$  will be recovered with the Basic OPSSR scheme.

The security proof still holds if all answers to oracle queries are independent for different values of  $\hat{m}$ . Therefore the functions  $E$  and  $E^{-1}$  need to take  $\hat{m}$  in their input. For better efficiency, a hash of  $\hat{m}$  is used.

In practice, OPSSR will use a collision free hash function  $H$  with  $2k$  bits of output and a keyed permutation  $E_k$  of blocks of size  $l$  with a key of size  $2k$ . The function  $E_k$  is modeled as an ideal cipher.

**Signature Generation.** The message is split  $m = \hat{m} \parallel \bar{m}$  with  $l - k$  bits in  $\bar{m}$ . Then  $h = H(\hat{m})$  and  $x = E_h^{-1}(\bar{m} \parallel \kappa)$  and  $s = F^{-1}(x)$  are computed. The signed message is  $\sigma = \hat{m} \parallel s$ .



**Signature Verification.** The signed message is split  $\sigma = \hat{m} \| s$  with  $s \in \mathcal{S}$ . Then  $x = F(s)$  and  $h = H(\hat{m})$  and  $\bar{m} \| v = E_h(x)$  are computed. The signature is valid if  $v = \kappa$ .

#### 4.4 RSA-OPSSR and Comparison with Other Schemes

**RSA-OPSSR.** With a goal of 80 bits of security and  $\log_2 q_S \simeq 48$ , OPSSR can be used for a proven (in the ideal cipher model) deterministic signature algorithm with 80 bits of message expansion with a 4096 bits RSA (or whatever is the modulus size for 128 bits of RSA security), or for a proven probabilistic signature algorithm with 128 bits of message expansion with 48 bits of salt and 1536 bits RSA (80 bits of RSA security).

**RSA-PSSR.** With a goal of 80 bits of security and  $\log_2 q_S \simeq 48$ , PSSR can be used for a proven (in the random oracle model) deterministic signature algorithm with 160 bits of message expansion with a 4096 bits RSA (or whatever is the modulus size for 128 bits of RSA security), or for a proven probabilistic signature algorithm with 208 bits of message expansion with 48 bits of salt and 1536 bits RSA (80 bits of RSA security).

**PVSSR or Naccache-Stern.** With a goal of 80 bits of security and  $\log_2 q_S \simeq 48$ , They can be used for a proven (in the generic group model) probabilistic signature algorithm based on 160 bits elliptic curve discrete logarithm and achieving 240 to 208 bits of message expansion.

## 5 Idealized Security Models and Multi-key Setting

### 5.1 The Multi-key Setting

Proofs of security for digital signature schemes only consider the case where the forger is able to ask signature queries for one public key, and has to make a valid signature for that public key.

However, it may be the case that computations done by the forger to attack one public key also help to attack another public key. Taking this into consideration is called the multi-key setting.

This consideration first appeared in a different form in the description of KCDSA for security against parameter manipulation [15, section 4.2].

Since the performance cost for having proofs of security against attacks in the multi-key setting is small, we believe that signature schemes should take this into account.

### 5.2 A Concrete Solution

To make the proof take the multi-key setting in account, one can make sure that all the components completely change if the public key changes.

For RSA-OPSSR, we have to meet the two following requirements:

- the best way to factor a bunch of RSA numbers is to factor separately each of them,
- the function  $E$ , in the idealized world, depends on the public key.

The first requirement does not depend on the padding and may not be met by the RSA bijection, because it may be possible to factor a bunch of RSA numbers faster than factoring them individually [7].<sup>2</sup>

To meet the second requirement we propose here a straightforward and simple improvement of the OPSSR scheme. The only change is that  $h = H(\hat{m}, pk)$ .

All other signature schemes proven secure in an idealized model can benefit from a similar improvement of their security. For example with RSA-PSS, it is sufficient to include the public key in the input of both hash functions  $H$  and  $G$ .

## 6 Discussion and Open Problems

### 6.1 Large Block Cipher

OPSSR with 4096 bits RSA needs a block cipher able to encrypt blocks of 512 bytes. No such block cipher has been widely studied. Using a deterministic mode of operation of a 8 or 16 byte block cipher is not a solution because it is not a valid implementation of the ideal cipher model.

Two research directions can be proposed.

- Is it possible to replace this ideal cipher with random oracles, for example with a sufficient number of Feistel rounds?
- How many rounds of the generalization of Rijndael that is based on 512 parallel S-boxes and an adequate MDS matrix are needed to have a secure cipher?

### 6.2 Optimal Trapdoor One-Way Permutations

Another drawback of using OPSSR with RSA is that even if the message expansion is small, the minimal length for a signed message is equal to the size  $l' = \lceil l \rceil$  of the RSA modulus. Optimal trapdoor one-way permutation have minimal input length and would minimize this value.

With an optimal trapdoor one-way (non random-self-reducible) permutation, i.e. that permutes  $l$  bits blocks with  $k' = l$  bits of security, (deterministic) OPSSR can be applied with  $l \simeq 2k$ . The minimal length for a signed message is  $2k$  and the message expansion is  $k$ .

With an optimal random-self-reducible trapdoor one-way permutation, i.e. that permutes  $l$  bits blocks with  $k' = l/2$  bits of security, (deterministic) OPSSR can be applied with  $l = 2k' = 2(k + \log_2 q_S)$ . The minimal length for a signed message is  $2k + 2\log_2 q_S$  and the message expansion is  $k$ . Randomized OPSSR

<sup>2</sup> This requirement is not met for schemes with security based on the hardness of the discrete logarithm in some fixed integer multiplicative group. The multi-key setting needs distinct groups for distinct public keys.

can also be applied with  $\#r = \log_2 q_S$  and  $l = 2k' = 2k$ . The minimal length for a signed message is  $2k + \log_2 q_S$  and the message expansion is  $k + \log_2 q_S$ .

But the problem of finding an explicit candidate for being an optimal (random-self-reducible) trapdoor one-way permutation is old and still unsolved.

### 6.3 Avoiding Idealized Security Models

The other important open problem is how to get rid of the idealized oracle models, which are the core of our proofs of security. Signature schemes based on chameleon hash functions or similar techniques cannot be an answer, because the information needed to commit to some hash has to be in the signed message, and will increase the message expansion.

### Acknowledgements

We would like to thank one anonymous referee for many interesting remarks that helped to improve this paper.

### References

1. M. Bellare and P. Rogaway. Optimal asymmetric encryption - how to encrypt with RSA. *Proc. Eurocrypt'94*, LNCS 950, pages 92-111, May 1994. Available from <http://www-cse.ucsd.edu/users/mihir/crypto-research-papers.html>.
2. M. Bellare and P. Rogaway. The exact security of digital signatures: how to sign with RSA and Rabin. *Proc. Eurocrypt'96*, LNCS 1070, pages 399-416, May 1996. Revised version available from <http://www-cse.ucsd.edu/users/mihir/crypto-research-papers.html>.
3. D. Boneh. Simplified OEAP for the RSA and Rabin functions. *Proc. Crypto'01*, LNCS 2139, pages 275-291, Aug. 2001. Available at <http://crypto.stanford.edu/~dabo/papers/saep.ps>.
4. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Proc. Asiacrypt'01*, LNCS 2248, pages 514-532, Dec. 2001. Available at <http://crypto.stanford.edu/~dabo/papers/weilsig.ps>.
5. D. Brown and D. Johnson. Formal Security Proofs for a Signature Scheme with Partial Message Recovery. 2000. Available at <http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-39.pdf>.
6. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Proc. STOC'98*, ACM, pages 209-218, May 1998. Available at <http://theory.lcs.mit.edu/~oded/rom.html>.
7. D. Coppersmith. Modifications of the Number Field Sieve. *Journal of Cryptology*, vol. 6, n. 3, pages 169-180, 1993.
8. J.-S. Coron. On the exact security of Full Domain Hash. *Proc. Crypto'00*, LNCS 1880, pages 229-235, Aug. 2000. Available at <http://www.eleves.ens.fr/home/coron/fdh.ps>.
9. J.-S. Coron. Optimal security proofs for PSS and other signature schemes. *Proc. Eurocrypt'02*, LNCS 2332, pages 272-287, May 2002. Available at <http://eprint.iacr.org/2001/062/>.

10. J.-S. Coron, M. Joye, D. Naccache and P. Paillier. Universal Padding Schemes for RSA. *Proc. Crypto'02*, LNCS, Aug. 2002. Available at <http://eprint.iacr.org/2002/115/>.
11. N. Courtois, M. Finiasz, and N. Sendrier. How to Achieve a McEliece-based Digital Signature Scheme. *Proc. Asiacrypt'01*, LNCS 2248, 157-174, Dec. 2001. Available at <http://www.minrank.org/mceliece/>.
12. N. Courtois, L. Goubin, and J. Patarin. Quartz, 128-bit long digital signatures. *Cryptographers' Track Rsa Conference 2001*, LNCS 2020, Apr. 2001. Available at <http://www.minrank.org/quartz/>.
13. A. Dent. Adapting the weaknesses of the Random Oracle model to the Generic Group model. To appear in *Asiacrypt'02*. Available at <http://eprint.iacr.org/2002/086/>.
14. J. Jonsson. An OAEP variant with a tight security proof. *Manuscript*, Mar. 2002. Available at <http://eprint.iacr.org/2002/034/>.
15. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. *Proc. Asiacrypt'98*, LNCS 1514, pages 175-186, Oct. 1998. Also available at <http://grouper.ieee.org/groups/1363/P1363a/PSSigs.html> as an IEEE P1363a submission.
16. K. Kobara and H. Imai. OAEP++ : A very simple way to apply OAEP to deterministic OW-CPA primitives. *Manuscript*, Aug. 2002. Available at <http://eprint.iacr.org/2002/130/>.
17. D. Naccache and J. Stern. Signing on a Postcard. *Proc. FC'00*, LNCS 1962, pages 121-135, Feb. 2000. Available at <http://grouper.ieee.org/groups/1363/Research/contributions/Postcard.ps>.
18. L. Pintsov and S. Vanstone. Postal revenue collection in the digital age. *Proc. FC'00*, LNCS 1962, pages 105-120, Feb. 2000. Available at <http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-43.ps>. Analysed in [5].
19. V. Shoup. OAEP Reconsidered. *Proc. Crypto'01*, LNCS 2139, pages 239-259, Aug. 2001. Available at <http://www.shoup.net/papers/oaep.pdf>.

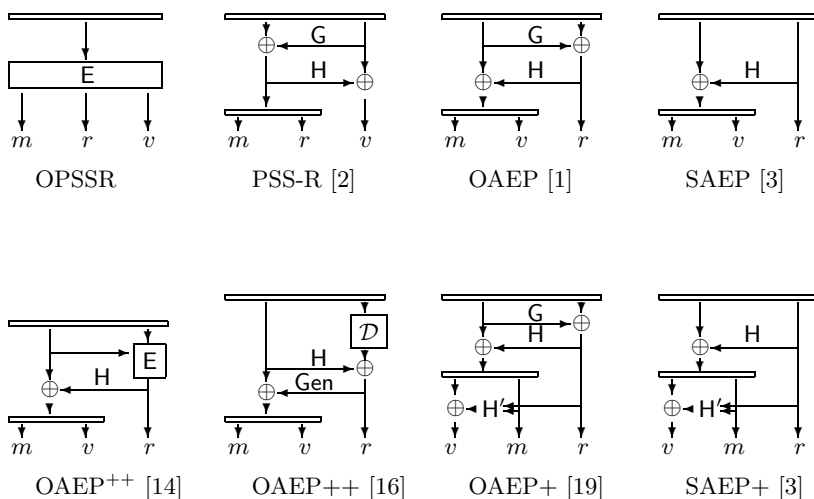
## A Comparison of OPSSR with Other Paddings

Many other paddings have been proposed. We show below the description of those paddings, when used for private decryption in an asymmetric encryption scheme or for public verification in a digital signature scheme. Their output is the message  $m$ , a random seed  $r$  and a validation value  $v$ . A non zero value for  $v$  leads to a rejection.

All these paddings have security proofs, where the internal components (the hash functions  $G$ ,  $H$  and  $H'$  and the encryption functions  $E$ ) are modeled as random oracles and ideal ciphers.

They are special implementations of OPSSR where the encryption function has a special form, but the security proof for OPSSR does not apply to this special form.

For example with PSS-R, if  $v$  is  $k$  bits long, then it is easy to find a collision  $H(m||r) = H(m'||r')$  in time  $2^{k/2}$ . If  $E$  is the corresponding encryption function for OPSSR (an unbalanced 2-rounds Feistel scheme based on  $G$  and  $H$ ), that means that if  $E^{-1}(m||r||v) = a||b$  is known, then the attacker can deduce that

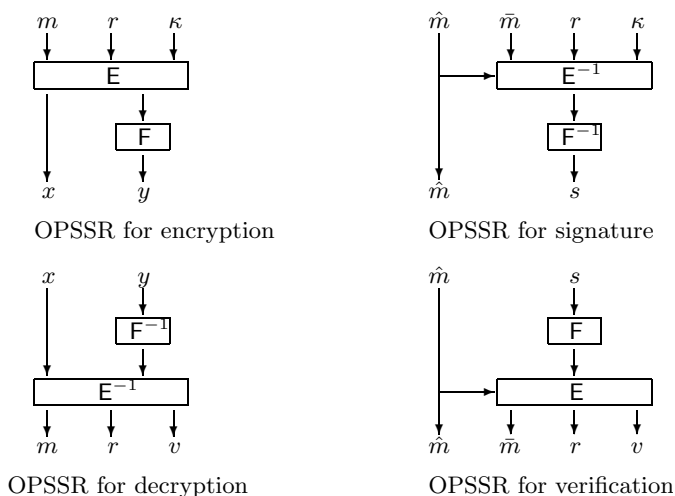


$E^{-1}(m' || r' || v) = a' || b$  where  $a' = a \oplus m || r \oplus m' || r'$ . This is incompatible with the ideal cipher model for  $E$ .

## B OPSSR Is an Optimal Universal Padding Scheme

**Basic OPSSR for Encryption.** This scheme can only encrypt messages of length  $l - k$ . It is built on a trapdoor one-way permutation  $F$  and a permutation  $E$  of blocks of size  $l$ .

The encryption of the message  $m$  is  $F \circ E(m || \kappa)$ . The decryption of the cipher  $c$  is  $m || v = E^{-1} \circ F^{-1}(c)$  and is rejected if  $v \neq \kappa$ .



To improve the tightness of the security proof, the scheme needs to be randomized. The encryption of  $m$  is  $F \circ E(m\|r\|\kappa)$  and the decryption  $m\|r\|v = E^{-1} \circ F^{-1}(c)$  is rejected if  $v \neq \kappa$ .

**OPSSR for Encryption.** To be able to encrypt arbitrary-length messages, one can use the same technique as Jonsson [14] and notice that the whole  $E(m\|r\|\kappa)$  does not need to be permuted with  $F$ . To encrypt  $m$  we compute  $x\|y = E(m\|r\|\kappa)$  and the cipher is  $c = x\|F(y)$ .

**Properties.** All properties of PSS described in [10] for a dual encryption + signature usage of the same public key are also valid for OPSSR. Moreover, the security reduction for the encryption scheme is as tight as for OAEP<sup>++</sup>. This can be proved with the technique from [14].

The main advantage of using OPSSR for encryption rather than these other paddings is that the message expansion is minimal, like it is the case for signature with OPSSR. The main disadvantage is that the encryption of a message of  $n$  bits with  $k$  bits of security and  $k$  bits of expansion needs a random permutation of blocks of  $n + k$  bits.

# The Provable Security of Graph-Based One-Time Signatures and Extensions to Algebraic Signature Schemes

Alejandro Hevia\* and Daniele Micciancio\*\*

Dept. of Computer Science & Engineering, University of California, San Diego,  
9500 Gilman Drive, La Jolla, California 92093, USA,  
{ahevia,daniele}@cs.ucsd.edu, www-cse.ucsd.edu/users/{ahevia,daniele}

**Abstract.** Essentially all known one-time signature schemes can be described as special instances of a general scheme suggested by Bleichenbacher and Maurer based on “graphs of one-way functions”. Bleichenbacher and Maurer thoroughly analyze graph based signatures from a combinatorial point of view, studying the graphs that result in the most efficient schemes (with respect to various efficiency measures, but focusing mostly on key generation time). However, they do not give a proof of security of their generic construction, and they leave open the problem of determining under what assumption security can be formally proved. In this paper we analyze graph based signatures from a security point of view and give sufficient conditions that allow to prove the security of the signature scheme in the standard complexity model (no random oracles). The techniques used to prove the security of graph based one-time signatures are then applied to the construction of a new class of algebraic signature schemes, i.e., schemes where signatures can be combined with a restricted set of operations.

## 1 Introduction

One-time signatures [Lam79] are digital signature schemes where the signer is restricted to sign a single document. They are interesting cryptographic primitives because they allow to solve many important cryptographic problems, and at the same time offer substantial efficiency advantages over regular digital signature schemes (cf. [RSA78,Sch90,GMR88,BM92]), especially with respect to signing, verification and key generation time. Applications of one time signatures include the design of regular signature schemes [Mer87,Mer90,BM92,DN94], on-line/off-line signatures [EGM96], digital signatures with forward security properties [BM99,AR00,MMM02], efficient broadcast authentication protocols [Per01[Roh99], network routing protocols [HPT97], and more. The first one-time signature scheme was proposed by Lamport [Lam79] and (in an interactive setting) by Rabin [Rab78]. The idea of the basic scheme of Lamport is very simple: given a

---

\* Supported in part by NSF grant CCR-0093029 and Mideplan Scholarship.

\*\* Supported in part by NSF Career Award CCR-0093029.

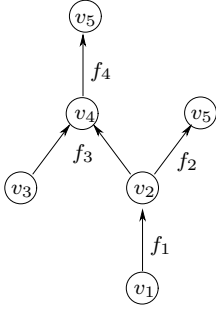
one-way function  $f$ , one selects two random strings  $x_0, x_1$  (which constitute the secret key), and publishes  $f(x_0), f(x_1)$ . Then, a single bit message  $b \in \{0, 1\}$  can be signed by revealing  $x_b$ . Verification is performed in the obvious way. Notice how the signing process is almost instantaneous, while verification only involves a single application of a one-way function. Key generation is almost as efficient, requiring only two applications of the one-way function.

Since Lamport’s original proposal, many extensions and improvements have been suggested [MM82, Mer82, Mer87, Vau92, BC93, EGM96, BM94, BM96b, BM96a, Per01]. The improvements usually involve iterating the application of the one-way function, or revealing multiple values as part of a signature. All these schemes (with the exception of Perrig’s) can be described as special instances of a general scheme suggested by Bleichenbacher and Maurer [BM94, BM96b, BM96a], based on the use of “graphs of one-way functions”. These are directed acyclic graphs or DAGs (see next section for a formal definition) with values associated to the vertices computed according to one-way functions associated to the edges (see Figure 1). Messages are signed by revealing the values for some of the vertices, and signatures verified using the publicly available one-way functions. As pointed out in [BM94, BM96b, BM96a] DAG-based one-time signatures schemes generalize and have potential advantages over schemes simply based on the iterated application of the one-way function (which correspond to graphs consisting of a collection of disjoint chains). Unfortunately, one-wayness does not seem a sufficiently strong assumption to guarantee the security of the graph based one time signature schemes. In fact, [BM94] and subsequent papers only study the combinatorial properties of the graphs, e.g., trying to maximize the size of the message space that can be signed using graphs with a predetermined number of vertices. The issue of determining sufficient security assumptions on the “one-way function”  $f$ , and proving the security of graph based signatures in the standard complexity model is left open in [BM94, BM96b, BM96a].

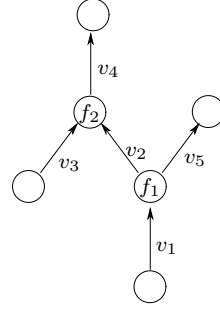
**OUR CONTRIBUTIONS:** In this paper we analyze the security of graph based signatures in order to put them on the firm grounds of the standard computational complexity security model. We show that under standard assumptions the security of graph based signatures can be formally proved. In order to achieve provable security, we adopt an approach in the definition of graph based signatures that is dual to the one used in [BM94]. Namely, instead of associating values to the nodes of a graph and functions to the edges, we propose to associate values to the edges and functions to the nodes (Figure 2 shows an example). Then, we prove that if the functions associated to the nodes are regular collision resistant (or simply universal one-way) hash functions and one-to-one pseudorandom generators, then the resulting one-time digital signature scheme is provably hard to break. These primitives can be built starting from any one-way permutation. The regularity and one-to-one properties can be relaxed assuming that the hash functions and pseudo-random generator only satisfy pseudorandomness and collision resistant properties.

An important byproduct of this work is the use of a hybrid argument in a novel way in our proof. Indeed, in order to prove the security of the signature





**Fig. 1.** DAG where values are associated to vertices and functions to edges (e.g.  $v_2 = f_1(v_1)$ ,  $v_4 = f_3(v_3, v_2)$ ,  $v_5 = f_4(v_4)$ ).



**Fig. 2.** DAG where values are associated to edges and functions to vertices (e.g.  $(v_2, v_5) = f_1(v_1)$ ,  $v_4 = f_2(v_3, v_2)$ ).

scheme, our analysis involves telling two distributions apart. However, a direct hybrid argument cannot be used because the number of hybrid distributions may be exponential on the security parameter. We show that by carefully setting a total order relation on the hybrids, we can combine them into a small (polynomial) number and the proof goes through. To the best of our knowledge this is a novel use of hybrid argument and may be of independent interest.

**EXTENSIONS:** Graph-based one-time signatures can be extended to instantiate a new type of signature scheme referred as *algebraic signatures*, originally suggested by Rivest [MR02]. An algebraic signature scheme is a signature scheme in which computing signatures of unseen messages is allowed in a restricted way. Associated to each algebraic signature scheme there is a set of functions  $\mathcal{O} = \{f_1, \dots, f_t\}$  (where each function  $f_i$  maps messages into messages). The fundamental property of algebraic signature schemes is that given signatures  $\text{sig}(m_1), \dots, \text{sig}(m_r)$  anyone can compute signature  $\text{sig}(f_i(m_1, \dots, m_r))$ . Clearly, algebraic signatures require the definition of a new notion of unforgeability. Namely, an algebraic signature scheme is secure if no adversary can efficiently compute signatures of messages that cannot be computed from  $m_1, \dots, m_r$  by applying the functions in  $\mathcal{O}$ . (See Section 6 for details). Micali and Rivest [MR02], and, recently, Bellare and Neven [BN02], presented constructions of *transitive signatures* which allow to sign edges in an undirected graph in such a way that computing signatures of the transitive closure of the signed edges does not require knowledge of the secret key. Similarly, Johnson et al. [JMSW02] studied several cases where the signing algorithm is homomorphic with respect to a binary operation  $f_i$ .

Building on graph-based one-time signature schemes we give explicit constructions for algebraic signatures on sets which support union and subset operations and also union and super-intersection operations<sup>1</sup>. We see graph-based

<sup>1</sup> The super-intersection of sets  $A$  and  $B$ , denoted  $A \odot B$ , is the collection of all sets  $S$  such that  $A \cap B \subseteq S \subseteq A \cup B$ .

algebraic signatures as an area that deserves further research, since it may lead to efficient and useful constructions.

## 2 Notation and Basic Definitions

In this section we review some definitions used throughout the paper. We start by recalling some standard definitions about cryptographic primitives and directed graphs.

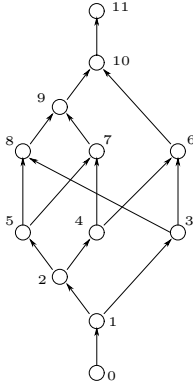
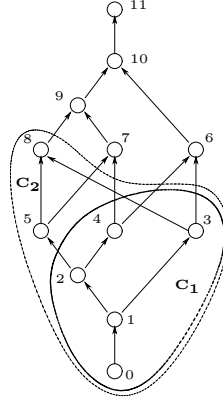
### 2.1 Cryptographic Primitives

We first recall the standard definition of security of signature schemes under chosen-message attacks (cf. [GMR88]) adapted to the case of one-time signature schemes. Then, we recall the (also standard) definitions of security of collision-resistant one-way hash functions (cf. [BR97]) and pseudorandom generators (cf. [BM84, Yao82]).

**ONE-TIME SIGNATURE SCHEME:** Formally, a signature scheme consists of three algorithms  $\Sigma = (\text{KG}, \text{Sig}, \text{Vf})$ . Given a security parameter  $k \in \mathbf{N}$ , the key generation algorithm  $\text{KG}(k)$  outputs a pair of public and private keys  $(pk, sk)$ ;  $\text{Sig}$  is the signing algorithm taking as input a key  $sk$  and a message  $m$ , and returning a signature  $\sigma$ ;  $\text{Vf}$  is the verification algorithm taking as input a key  $pk$ , a message  $m$  and a signature  $\sigma$ , and returning a boolean decision. The signing algorithm may be randomized but the verification algorithm is usually deterministic. It is required that valid signatures are always accepted. A one-time signature scheme is secure against existential forgery in a one-chosen-message attack if no computationally bounded adversary (*forgery*), after obtaining the signature of a single message of his choice, can output a (different) message and a corresponding valid signature, except with negligible probability.

**COLLISION-RESISTANT HASH FUNCTIONS:** Let  $\mathcal{H}$  be a family of functions. An individual element in  $\mathcal{H}$  is function  $H: R^2 \rightarrow R$ , for some fix set  $R$ . The family  $\mathcal{H}$  is said to be collision-resistant if, for  $H$  randomly chosen in  $\mathcal{H}$ , any computationally bounded adversary (*collision-finder*) can not find two different messages  $m$  and  $m'$  that map by  $H$  to the same value, except with negligible probability. Furthermore, we say  $\mathcal{H}$  is *regular* if it satisfies  $\Pr \left[ H(X) = y : X \xleftarrow{R} R^2 \right] = \Pr \left[ Y = y : Y \xleftarrow{R} R \right]$  for all  $y \in R$ , and all  $H \in \mathcal{H}$ .

**PSEUDORANDOM GENERATORS:** Let  $G: R \rightarrow R^2$  be a deterministic function.  $G$  is a pseudorandom generator if it no computationally bounded adversary (*distinguisher*) can tell apart the output of  $G(x)$  on a random input  $x$  from a truly random value on  $R^2$  with non-negligible probability. Also, a pseudorandom generator  $G$  is *one-to-one* if there is no pair of distinct inputs  $x, x' \in R$ , that produce the same output on  $G$ .

Fig. 3. Example of a DAG  $\mathcal{G}$ .Fig. 4. Two cuts  $C_1 \subseteq C_2$  in  $\mathcal{G}$ .

## 2.2 Graphs

A *directed graph* is a pair  $(V, E)$  where  $V$  is a finite set of *vertices* and  $E \subseteq V \times V$  is the set of *edges*. A *path* of length  $\ell \geq 0$  from  $v_0$  to  $v_\ell$  in  $G$  is a sequence of vertices  $p = (v_0, \dots, v_\ell)$  such that  $(v_{i-1}, v_i) \in E$  for all  $i = 1, \dots, \ell$ . If such a path exists, we say that  $v_0$  is a *predecessor* of  $v_\ell$  and  $v_\ell$  is a *successor* of  $v_0$ . The sets of predecessors and successors of  $v$  are denoted  $\text{Pred}(v)$  and  $\text{Succ}(v)$ , respectively. A set of vertices  $S$  is *predecessor closed* if  $\text{Pred}(v) \subseteq S$  for all  $v \in S$ . Similarly,  $S$  is *successor closed* if  $\text{Succ}(v) \subseteq S$  for all  $v \in S$ . A *cycle* is a path  $(v_0, \dots, v_\ell)$  of length  $\ell \geq 1$  such that  $v_0 = v_\ell$ . A *directed acyclic graph* (DAG) is a directed graph with no cycles.

The *indegree* of a vertex  $v$  is the number of edges  $(v', v) \in E$  pointing to  $v$ , the *outdegree* is the number of edges  $(v, v') \in E$  departing from  $v$ , and the *total degree* is the sum of the indegree and the outdegree. Vertices with indegree 0 are called *sources*, and vertices with outdegree 0 are called *sinks*. Vertices that are neither sources nor sinks are called *internal vertices*. For simplicity, in this paper we only considers DAGs with a single source  $v_\perp$  with outdegree 1, a single sink  $v_\top$  with indegree 1, and  $n > 0$  internal nodes with total degree 3. For such graphs, there are only two kind of internal vertices: *expansion vertices* with indegree 1 and outdegree 2, and *compression vertices* with indegree 2 and outdegree 1. So, the sets of vertices of our graphs can be partitioned as  $V = V_G \cup V_H \cup \{v_\perp, v_\top\}$ , where  $V_G$  are the expansion vertices and  $V_H$  the compression vertices. We also fix a total order relation  $(V_G, \leq)$  that extends the partial order defined over  $V_G$  by the predecessor relation.

An example of DAG is depicted in Figure 3. Vertex 0 is the source, vertex 11 the sink,  $V_H = \{1, 2, 3, 4, 5\}$  are compression vertices, and  $V_G = \{6, 7, 8, 9, 10\}$  are expansion vertices.

A *cut* in a graph  $(V, E)$  is a nontrivial partition  $C = (S, \bar{S})$  of the vertices such that  $S$  is predecessor closed (or, equivalently,  $\bar{S}$  is successor closed). The set of cuts in a graph  $(V, E)$  is denoted  $\text{Cuts}(V, E)$ , and it forms a partial order

where  $(S, \bar{S}) \sqsubseteq (S', \bar{S}')$  if and only if  $S \subseteq S'$  (or, equivalently,  $\bar{S} \supseteq \bar{S}'$ ). Notice that since  $(S, \bar{S})$  is nontrivial (i.e., both  $S$  and  $\bar{S}$  are not empty), and  $S, \bar{S}$  are predecessor and successor closed, it is always the case that  $v_\perp \in S$  and  $v_\top \in \bar{S}$ . Therefore, a cut can be implicitly represented by a single set of vertices  $S$  with the convention that if  $v_\perp \in S$  then  $(S)$  represents  $(S, V \setminus S)$ , while if  $v_\top \in S$  then  $(S)$  represents  $(V \setminus S, S)$ . For any cut  $C$ , the component of  $C$  containing  $v_\perp$  (resp.  $v_\top$ ) is denoted  $S(C)$  (resp.  $\bar{S}(C)$ ).

An edge  $e = (u, v)$  *crosses* a cut  $C = (S, \bar{S})$  if  $u \in S$  and  $v \in \bar{S}$ . The set of edges crossing  $C$  is denoted  $\text{Edges}(C) = E \cap (S \times \bar{S})$ . We consider graphs where each edge is labeled with an element from some set  $R$ . The labels associated to the edges are not totally independent, but must satisfy certain constraints. Let  $G: R \rightarrow R^2$  and  $H: R^2 \rightarrow R$  be two arbitrary functions. (Later on, we will instantiate  $G$  with a pseudorandom generator and  $H$  with a collision resistant hash function.) A labeling is a partial function  $\lambda$  from  $E$  to  $R$ , i.e., a function  $\lambda: T \rightarrow R$  where  $T \subseteq E$ . The domain  $T$  of the labeling is denoted  $\text{dom}(\lambda)$ . We say that  $\lambda$  is consistent (with respect to functions  $G$  and  $H$ ) if values are computed according to functions  $G$  and  $H$ , i.e.,

- for every expansion vertex with incoming edge  $e_0 \in \text{dom}(\lambda)$  and outgoing edges  $e_1, e_2 \in \text{dom}(\lambda)$ ,  $G(\lambda(e_0)) = (\lambda(e_1), \lambda(e_2))$ .
- for every compression vertex with incoming edges  $e_0, e_1 \in \text{dom}(\lambda)$  and outgoing edge  $e_2 \in \text{dom}(\lambda)$ ,  $\lambda(e_2) = H(\lambda(e_0), \lambda(e_1))$ .

We are interested in labeling functions defined over cuts. A *labeled cut* is a labeling function  $\sigma$  such that  $\text{dom}(\sigma)$  is the set of edges of a cut, i.e.,  $\text{dom}(\sigma) = \text{Edges}(C)$  for some  $C \in \text{Cuts}(V, E)$ . If  $\sigma$  is a labeling with domain  $\text{Edges}(C)$  then we write  $\sigma: C$ . Similarly, we denote as  $\{\sigma: C\}$  the set of all labellings with domain  $\text{Edges}(C)$ . Notice that any function  $\sigma: \text{Edges}(C) \rightarrow R$  is consistent, i.e., the edges of a cut can be labeled independently. Any labeled cut  $\sigma: C$  can be uniquely extended to a consistent labeling defined over all edges ending in  $\bar{S}(C)$ .

**Proposition 1.** *For any directed acyclic graph  $(V, E)$ , cut  $C \in \text{Cuts}(V, E)$  and labeling  $\sigma: \text{Edges}(C) \rightarrow R$ , there exists a unique labeling, denoted  $[\sigma]$ , such that*

- (1)  $\text{dom}([\sigma]) = E \cap (V \times \bar{S}(C))$
- (2)  $[\sigma]$  is consistent, and
- (3)  $[\sigma](v) = \sigma(v)$  for all  $v \in \text{Edges}(C)$ .

*Moreover,  $[\sigma]$  can be efficiently computed from  $\sigma$ .*

Notice that for any two cuts  $C_1 \sqsubseteq C_2$ , the set  $\text{Edges}(C_2)$  is contained in  $V \times \bar{S}(C_1)$ . Therefore, given a labeled cut  $\sigma_1: C_1$  and a cut  $C_2$  such that  $C_1 \sqsubseteq C_2$ , we can define a labeled cut  $\sigma_2: C_2$  by restricting the domain of  $[\sigma_1]$  to  $\text{Edges}(C_2)$ .

**Definition 1.** *For any ordered pair of cuts  $C_1 \sqsubseteq C_2$ , we define a corresponding projection operation  $\Pi_{C_2}^{C_1}$  (or, simply,  $\Pi_{C_2}$  when  $C_1$  is clear from the context) that maps any labeled cut  $\sigma_1: C_1$  to a corresponding labeled cut  $\sigma_2: C_2$  obtained by first extending  $\sigma_1$  to  $[\sigma_1]$ , and then restricting the domain of  $[\sigma_1]$  to the set  $\text{Edges}(C_2)$ .*

Notice that if  $C_1 = (S_1, \bar{S}_1)$  and  $C_2 = (S_2, \bar{S}_2)$ , then  $\sigma_2 = \Pi_{C_2}(\sigma_1)$  can be computed from  $\sigma_1$  with at most  $|S_2 \setminus S_1|$  applications of functions  $G$  and  $H$ .

*Example 1.* Figure 4 depicts two example cuts  $S(C_1) = \{0, 1, 2, 3, 4\}$  with  $\text{Edges}(C_1) = \{(2, 5), (4, 7), (4, 6), (3, 6), (3, 8)\}$ , and  $S(C_2) = \{0, 1, 2, 3, 4, 5, 8\}$  with  $\text{Edges}(C_2) = \{(8, 9), (5, 7), (4, 7), (4, 6), (3, 6)\}$ . As a toy example, consider  $R = \mathbb{Z}_{10}$ ,  $H(x, y) \stackrel{\text{def}}{=} x + y$ , and  $G(x) \stackrel{\text{def}}{=} (x, x)$ . If we choose  $\{((2, 5), 3), ((4, 7), 9), ((4, 6), 5), ((3, 6), 2), ((3, 8), 8)\}$  as a labeled cut  $\sigma: C_1$  in  $\mathcal{G}$ , then it is easy to check that the labeled cut defined by  $\Pi_{C_1}^{C_2}(\sigma)$  (the consistent extension of  $C_1$  onto  $C_2$ ) is  $\{((8, 9), 1), ((5, 7), 3), ((4, 7), 9), ((4, 6), 5), ((3, 6), 2)\}$ .

### 3 The GBOTS Construction

A graph based one-time signature (GBOTS) scheme is specified by a directed acyclic graph  $(V, E)$ , a function  $\mu: \mathcal{M} \rightarrow \text{Cuts}(V, E)$  from a message space  $\mathcal{M}$  to the set of cuts of the graph, a length doubling function  $G: R \rightarrow R^2$  and a family  $\mathcal{H}$  of length halving functions  $H: R^2 \rightarrow R$ . Function  $\mu$  must satisfy the security property that if  $m \neq m'$ , then the cuts  $\mu(m)$  and  $\mu(m')$  are incomparable, i.e., neither  $\mu(m) \subseteq \mu(m')$  nor  $\mu(m') \subseteq \mu(m)$ . In particular, function  $\mu$  is injective. Examples of such functions are presented in [BM96b, BM96a].

The secret key of a GBOTS scheme consists of a labeled cut  $\sigma_\perp: \{v_\perp\}$  and a hash function  $H \in \mathcal{H}$ , both chosen uniformly at random. The corresponding public key is given by function  $H$  and the labeled cut  $\sigma_\top = \Pi_{\{v_\top\}}(\sigma_\perp)$ . A signature for a message  $m \in \mathcal{M}$  is a labeled cut  $\sigma: \mu(m)$ . Message  $m$  is signed using secret key  $(H, \sigma_\perp)$  setting  $\sigma = \Pi_{\mu(m)}(\sigma_\perp)$ . A message signature pair  $(m, \sigma: \mu(m))$  is verified using public key  $(H, \sigma_\top)$  checking that  $\Pi_{\{v_\top\}}(\sigma) = \sigma_\top$ . A formal specification of the GBOTS scheme is given in Figure 5.

Algorithm $\text{KG}(1^k)$	Algorithm $\text{Sig}(sk, m)$	Algorithm $\text{Vf}(pk, m, \sigma)$
$H \xleftarrow{R} \mathcal{H}, \sigma_\perp \xleftarrow{R} \{\sigma: \{v_\perp\}\}$	<b>parse</b> $sk$ as $(H, \sigma_\perp)$	<b>parse</b> $pk$ as $(H, \sigma_\top)$
$\sigma_\top \leftarrow \Pi_{\{v_\top\}}(\sigma_\perp)$	$\sigma \leftarrow \Pi_{\mu(m)}(\sigma_\perp)$	<b>if</b> $\Pi_{\{v_\top\}}(\sigma) = \sigma_\top$
$pk \leftarrow (H, \sigma_\top), sk \leftarrow (H, \sigma_\perp)$	<b>return</b> $\sigma: \mu(m)$	<b>return</b> 1
<b>return</b> $(pk, sk)$		<b>else return</b> 0

**Fig. 5.** Key Generation, Signing and Verification algorithms for GBOTS scheme.

### 4 The Reduction

In this section we relate the security of GBOTS to the security of the underlying pseudorandom generator  $G$  and family of hash functions  $\mathcal{H}$ . Formally, we show how a forger adversary  $\mathcal{F}$  that successfully attacks the one-time signature scheme, can be used to build efficient procedures to successfully attack  $G$  and  $\mathcal{H}$  as follows: an inverter algorithm  $\mathcal{I}_H$  that attempts to invert a randomly chosen

function  $H \in \mathcal{H}$ ; an inverter algorithm  $\mathcal{I}_G$  that attempts to invert function  $G$ ; a collision finder algorithm  $\mathcal{C}_H$  that on input  $H \in \mathcal{H}$  attempts to find a collision to  $H$ , and a distinguisher  $\mathcal{D}_G$  that attempts to tell random strings and pseudorandom strings apart.

None of the adversaries  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$  is individually guaranteed to work, but we can bound the success probability of the forger  $\mathcal{F}$  as a function of the combined success probabilities of  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$ . So, if  $G, \mathcal{H}$  are cryptographically secure, then the GBOTS scheme is secure. In the rest of this section we show how to build  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$  given black box access to the forger  $\mathcal{F}$ . The success probabilities of these adversaries are analyzed in the following section.

Adversaries  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$  all use the forger  $\mathcal{F}$  in a specific way, common to all four of them. So, we describe this general procedure  $\mathcal{A}$  first. This procedure takes as input a hash function  $H$ , a node  $v$ , and a labeling  $\sigma_v: \text{Pred}(v)$ . The task is, given oracle access to the forger algorithm, compute a labeling  $\sigma'_v: \text{Succ}(v)$ . In other words,  $\mathcal{A}$  gets as input a labeling of the smallest cut containing  $v$ , and tries to output a labeling for the biggest cut not containing  $v$  (where biggest and smallest refer to the  $\sqsubseteq$  ordering relation).

Procedure  $\mathcal{A}(H, v, \sigma_v)$  operates as follows:

1. Compute  $\sigma_\top = \Pi_{\{v_\top\}}(\sigma_v)$ .
2. Run  $\mathcal{F}$  on input  $pk = (H, \sigma_\top)$ .
3. Let  $m \in \mathcal{M}$  be the message output by  $\mathcal{F}$ . If  $v \notin \mu(m)$ , then abort. Otherwise, compute  $\sigma_m = \Pi_{\mu(m)}(\sigma_v)$  and continue to the next step.
4. Run  $\mathcal{F}$  on input  $\sigma_m$  to get a forgery  $m', \sigma'$ . We assume, without loss of generality, that  $\mathcal{F}$  always outputs a valid message-signature pair, i.e.,  $\Pi_{\{v_\top\}}(\sigma') = \sigma_\top$ . If  $\mathcal{F}$  cannot forge a signature, then it outputs  $(m, \sigma_m)$ .
5. If  $v \in \mu(m')$  then abort. Otherwise, compute and output  $\sigma'_v = \Pi_{\text{Succ}(v)}(\sigma')$ .

A few remarks follow. First, for any vertex  $v$ ,  $\text{Pred}(v) \sqsubseteq \{v_\top\}$ , so the projection operation in step 1 can always be performed. This produces a pair  $pk = (H, \sigma_\top)$  which is similar, but not necessarily identically distributed, to a public key. In step 3, if  $v \in \mu(m)$ , then  $\text{Pred}(v) \subseteq \mu(m)$  because cut  $\mu(m)$  is closed. So, unless execution is aborted,  $\text{Pred}(v) \sqsubseteq \mu(m)$  and  $\sigma_m$  can be computed from  $\sigma_v$ . Similarly, in step 5, if execution does not abort,  $v \notin \mu(m')$  and  $\mu(m') \sqsubseteq \text{Succ}(v)$ . So,  $\sigma'_v$  can be computed from  $\sigma'$ . Therefore,  $\mathcal{A}$  always either aborts or it succeeds, i.e., it outputs a cut  $\sigma'_v: \text{Succ}(v)$  such that  $\Pi_{\{v_\top\}}(\sigma'_v) = \sigma_\top$ . We use  $\mathcal{A}$  to define  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H$  and  $\mathcal{D}_G$ .

#### 4.1 Inverting $H$

Algorithm  $\mathcal{I}_H$  on input a hash function  $H$  and target value  $y \in R$ , chooses one vertex  $v \in V_H$  at random, and selects  $\sigma_v$  uniformly at random among all labeled cuts  $\sigma: \text{Pred}(v)$  such that  $\sigma(e) = y$ , where  $e$  is the only edge departing from  $v$ . Then algorithm  $\mathcal{I}_H$  calls  $\mathcal{A}(H, v, \sigma_v)$ . If  $\mathcal{A}$  aborts, also  $\mathcal{I}_H$  aborts. Otherwise, let  $\sigma'_v: \text{Succ}(v)$  be the signature output by  $\mathcal{A}$ . The output of  $\mathcal{I}_H$  is  $\sigma'_v(e_0); \sigma'_v(e_1)$ , where  $e_0, e_1$  are the edges pointing to  $v$ .

We remark that  $\mathcal{I}_H$  may either abort, terminate successfully with a pre-image of  $y$  under  $H$ , or fail, i.e., terminate without aborting, but with an output value  $x_0; x_1$  such that  $H(x_0; x_1) \neq y$ . The distinction between aborting execution and failure to invert will be used in the analysis.

## 4.2 Inverting $G$

The algorithm to invert  $G$  is similar to  $\mathcal{I}_H$ .  $\mathcal{I}_G$  on input a target value  $(x_1; x_2) \in R^2$ , chooses  $H \in \mathcal{H}$  uniformly at random, picks one vertex  $v \in V_G$ , and selects  $\sigma_v$  uniformly among all labeled cuts  $\sigma: \text{Pred}(v)$  such that  $\sigma(e_1) = x_1$  and  $\sigma(e_2) = x_2$ , where  $e_1, e_2$  are the edges departing from  $v$ . Then it calls  $\mathcal{A}(H, v, \sigma_v)$ . If  $\mathcal{A}$  aborts, also  $\mathcal{I}_G$  aborts. Otherwise, let  $\sigma'_v: \text{Succ}(v)$  be the signature output by  $\mathcal{A}$ . The output of  $\mathcal{I}_G$  is  $\sigma'_v(e_0)$  where  $e_0$  is the edge pointing to  $v$ . As for  $\mathcal{I}_H$ , inverter  $\mathcal{I}_G$  can either abort, terminate successfully, or fail.

## 4.3 Finding Collisions

In order to describe the collision finder algorithm we need the following lemma. The proof is simple and can be seen in the full version of this paper [HM02]. The proof uses the assumption that  $G$  is one-to-one.

**Lemma 1.** *For any cut  $C \in \text{Cuts}(V, E)$ , and labellings  $\sigma: C$  and  $\sigma': C$ , if  $\sigma \neq \sigma'$  and  $\Pi_{\{v_\top\}}(\sigma) = \Pi_{\{v_\top\}}(\sigma')$ , then there exists a compression node  $v$  not in  $C$  with incoming edges  $e_0, e_1$  such that  $([\sigma](e_0), [\sigma](e_1))$  and  $([\sigma'](e_0), [\sigma'](e_1))$  form a collision, i.e.,  $H([\sigma](e_0), [\sigma](e_1)) = H([\sigma'](e_0), [\sigma'](e_1))$  and  $[\sigma](e_i) \neq [\sigma'](e_i)$  for some  $i \in \{0, 1\}$ .*

The collision finder  $\mathcal{C}_H$  takes as input a hash function  $H$ , and selects a vertex  $v \in V_G \cup V_H$  uniformly at random. Notice that  $v \in V_G$  and  $v \in V_H$  happen with the same probability because  $V_G$  and  $V_H$  have the same size. The rest of the collision finder algorithm is similar to  $\mathcal{I}_G$  or  $\mathcal{I}_H$ , depending on whether  $v \in V_G$  or  $v \in V_H$ .

If  $v \in V_G$ , then  $\mathcal{C}_H$  chooses  $x \in R$  uniformly at random, computes  $(y_1; y_2) = G(x)$ , and picks  $\sigma_v$  uniformly at random among all labeled cuts  $\sigma: \text{Pred}(v)$  such that  $\sigma(e_1) = y_1$  and  $\sigma(e_2) = y_2$ , where  $e_1, e_2$  are the edges departing from  $v$ . Then it calls  $\mathcal{A}(H, v, \sigma_v)$ . If  $\mathcal{A}$  aborts, also  $\mathcal{C}_H$  aborts. Otherwise, let  $\sigma'_v: \text{Succ}(v)$  be the signature output by  $\mathcal{A}$ , and consider the cut  $\text{Succ}(v) \setminus \{v\}$ . Notice that  $\text{Succ}(v) \sqsubseteq \text{Succ}(v) \setminus \{v\}$  and  $\text{Pred}(v) \sqsubseteq \text{Succ}(v) \setminus \{v\}$ . Therefore, we can compute two labeling  $\sigma = \Pi_{\text{Succ}(v) \setminus \{v\}}(\sigma_v)$  and  $\sigma' = \Pi_{\text{Succ}(v) \setminus \{v\}}(\sigma'_v)$ . If  $\sigma \neq \sigma'$ , then compute a collision from  $\sigma$  and  $\sigma'$  using Lemma 1.

If  $v \in V_H$ , then  $\mathcal{C}_H$  chooses  $x_0, x_1 \in R$  uniformly at random, compute  $x_2 = H(x_0, x_1)$ , and pick  $\sigma_v$  uniformly at random among all labeled cuts  $\sigma: \text{Pred}(v)$  such that  $\sigma(e_2) = y_2$ . It then call  $\mathcal{A}(H, v, \sigma_v)$ . If  $\mathcal{A}$  aborts, also  $\mathcal{I}_H$  aborts. Otherwise, let  $\sigma'_v: \text{Succ}(v)$  be the signature output by  $\mathcal{A}$ , and consider the cut  $\text{Succ}(v) \setminus \{v\}$ . As before,  $\text{Succ}(v) \sqsubseteq \text{Succ}(v) \setminus \{v\}$  and  $\text{Pred}(v) \sqsubseteq \text{Succ}(v) \setminus \{v\}$ . Therefore, we can compute two labeling  $\sigma = \Pi_{\text{Succ}(v) \setminus \{v\}}(\sigma_v)$  and  $\sigma' = \Pi_{\text{Succ}(v) \setminus \{v\}}(\sigma'_v)$ . If  $\sigma \neq \sigma'$ , then compute a collision from  $\sigma$  and  $\sigma'$  using Lemma 1.

#### 4.4 Distinguishing $G$

Finally we describe a possible distinguisher for  $G$ . On input  $x_1, x_2 \in R^2$ ,  $\mathcal{D}_G$  picks a random vertex  $v \in V$  and a hash function  $H \in \mathcal{H}$ . This time vertex  $v$  is not selected with uniform probability, but with probability proportional to  $|V_G \cap (\text{Pred}(v) \setminus \{v\})|$ . Then  $\mathcal{D}_G$  chooses a node  $u \in V_G \cap (\text{Pred}(v) \setminus \{v\})$  uniformly at random, and computes  $\sigma_v$  as follows. Let  $\{\sigma: \cup_{u' \leq u} \text{Pred}(u')\}$  denote the set of all labellings defined over the union of cuts  $\text{Pred}(u')$  for all expansion vertices  $u' \leq u$  in the predecessor set of  $v$  but not including  $v$ ; in other words, it denotes the union of cuts  $\text{Pred}(u')$  such that  $u' \leq u$ , and  $u' \in V_G \cap (\text{Pred}(v) \setminus \{v\})$ . In this union, each labeling satisfies  $\sigma(e_1); \sigma(e_2) = x_1; x_2$ , where  $e_1, e_2$  are the edges departing from  $u$ . Distinguisher  $\mathcal{D}_G$  selects  $\sigma_u$  uniformly at random in  $\{\sigma: \cup_{u' \leq u} \text{Pred}(u')\}$ , and computes  $\sigma_v = \Pi_{\text{Pred}(v)}(\sigma_u)$ . Notice that for all  $u'$  predecessor of  $v$ ,  $\text{Pred}(u') \subset \text{Pred}(v)$ , and the labeled cut  $\sigma_v$  can be computed from  $\sigma_u$ .

Procedure  $\mathcal{A}$  is run on input  $H, v, \sigma_v$ . If  $\mathcal{A}$  aborts then  $\mathcal{D}_G$  outputs “random”, while if  $\mathcal{A}$  does not abort  $\mathcal{D}_G$  outputs “pseudorandom”.

### 5 Analysis

In this section we relate the success probability of the forger algorithm  $\mathcal{F}$  to the success probability of attacks to  $G$  and  $H$ . The following result states that if  $G$  is a one-to-one pseudorandom generator and  $\mathcal{H}$  is a regular collision-resistant hash function family then the GBOTS scheme is existentially secure under one-chosen-message attack.

**Theorem 1.** *Let  $(V, E)$  be a directed acyclic graph,  $G$  a one-to-one pseudorandom generator, and  $\mathcal{H}$  a regular collision resistant family of hash functions, and consider the corresponding GBOTS scheme. Let  $\mathcal{F}$  be a forger that succeeds with probability  $\delta$ . Then  $\delta \leq (\alpha\epsilon_D + \epsilon_C + \epsilon_G + \epsilon_H)n$  where  $\alpha \leq n$  is the average number of  $V_G$  predecessors of a random vertex in the graph and  $\epsilon_G, \epsilon_H, \epsilon_C, \epsilon_D$  are the success probabilities (or advantage) of adversaries  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$  as defined in the previous section.*

In order to prove the result, we first show that the success probability of the adversaries  $\mathcal{I}_G, \mathcal{I}_H$  and  $\mathcal{C}_H$  is tightly related to the aborting probability of procedure  $\mathcal{A}$ , when called on randomly chosen inputs. We make this statement more precise below. First, we need some notation.

A labeled cut  $\sigma$  is said to be *consistent with*  $(v, y) \in V \times (R^2 \cup R)$  if one of two cases hold: **(a)** if  $v \in V_G$  and  $y = y_1; y_2 \in R^2$  then  $\sigma(e_1) = y_1$  and  $\sigma(e_2) = y_2$  where  $e_1$  and  $e_2$  are the edges departing from  $v$ , or **(b)** if  $v \in V_H$  and  $y \in R$  then  $\sigma(e) = y$  where  $e$  is the only edge departing from  $v$ . The set of all labeled cuts  $\sigma$  consistent with  $(v, y)$  is denoted  $\{\sigma: \text{Pred}(v)_y\}$ . In particular, if either  $v \in V_G$  and  $y = G(x)$  for  $x \in R$  chosen uniformly at random, or  $v \in V_H$  and  $y = H(x_1; x_2)$  for  $x_1; x_2 \in R^2$  chosen uniformly at random, the set  $\{\sigma: \text{Pred}(v)_y\}$  is denoted  $\{\sigma: \text{Pred}(v)_{H/G(\cdot)}\}$ .



Consider the following experiment. First, we choose a vertex  $v \in V_H \cup V_G$ , a hash function  $H \in \mathcal{H}$  and a labeled cut  $\sigma_v \in \{\sigma: \text{Pred}(v)_{H/G(\cdot)}\}$  uniformly at random. Then we call procedure  $\mathcal{A}$  on input  $(H, v, \sigma_v)$ . (For simplicity's sake, when clear from the context, we use  $\mathcal{A}(\cdot)$  to denote  $\mathcal{A}(H, v, \sigma_v)$ ). Let **NoAbort** denote the event that  $\mathcal{A}$  does not abort in this experiment. The following lemma shows that the combined success probability of adversaries  $\mathcal{I}_G$ ,  $\mathcal{I}_H$  and  $\mathcal{C}_H$  is equal to the probability of the event **NoAbort**.

**Lemma 2.** *Let  $\epsilon_H$ ,  $\epsilon_G$  and  $\epsilon_C$  the advantages of adversaries  $\mathcal{C}_H$ ,  $\mathcal{I}_G$  and  $\mathcal{I}_H$ . Let **NoAbort** be the event as described above. Then  $\epsilon_H + \epsilon_G + \epsilon_C = \Pr[\text{NoAbort}]$*

*Proof.* We analyze the success probability of adversaries  $\mathcal{I}_H$ ,  $\mathcal{I}_G$  and  $\mathcal{C}_H$  in turn. First, the success probability  $\epsilon_H$  of adversary  $\mathcal{I}_H$  is the probability that, for  $x_1; x_2 \in R^2$  and  $H \in \mathcal{H}$  uniformly chosen at random,  $H(\mathcal{I}_H(H, H(x_1; x_2))) = H(x_1; x_2)$ , that is, that  $\mathcal{I}_H$  returns a pre-image of  $H(x_1; x_2)$  for a random domain point  $x_1; x_2$ . For  $X \in \{H, G\}$ , let  $\Pr_X[E]$  denote the probability of event  $E$  when  $H \in \mathcal{H}$ ,  $v \in V_X$  and  $\sigma_v \in \{\sigma: \text{Pred}(v)_{H/G(\cdot)}\}$  are chosen uniformly at random. Then

$$\begin{aligned} \epsilon_H &= \Pr_H[H(x') = H(x), x' \leftarrow \mathcal{A}(H, v, \sigma_v), x' \neq \text{abort}] \\ &= (1 - \Pr_H[H(x') \neq H(x) \mid \mathcal{A}(\cdot) \neq \text{abort}]) \cdot \Pr_H[\mathcal{A}(\cdot) \neq \text{abort}] \end{aligned}$$

Similarly, for adversary  $\mathcal{I}_G$  we have

$$\epsilon_G = (1 - \Pr_G[G(x') \neq G(x) \mid \mathcal{A}(\cdot) \neq \text{abort}]) \cdot \Pr_G[\mathcal{A}(\cdot) \neq \text{abort}]$$

Lastly, recall that Adversary  $\mathcal{C}_H$  is successful if, after running  $\mathcal{A}$  on a randomly chosen  $v \in V_H \cup V_G$ , either  $G(x) \neq G(x')$  if  $v \in V_G$  or  $H(x) \neq H(x')$  if  $v \in V_H$ . Thus,

$$\begin{aligned} \epsilon_C &= \frac{1}{2} \cdot (\Pr_H[H(x') \neq H(x) \mid x' \leftarrow \mathcal{A}(\cdot), x' \neq \text{abort}] \cdot \Pr_H[\mathcal{A}(\cdot) \neq \text{abort}] + \\ &\quad \Pr_G[G(x') \neq G(x) \mid \mathcal{A}(\cdot) \neq \text{abort}] \cdot \Pr_G[\mathcal{A}(\cdot) \neq \text{abort}]) \end{aligned}$$

Combining the above results and using that  $|V_H| = |V_G|$  the result follows.

As a second step toward proving Theorem 1, next lemma shows that the success probability of the distinguisher  $\mathcal{D}_G$  is related to the difference between forger's success probability and the probability that procedure  $\mathcal{A}$  does not abort (in the experiment described in the previous lemma).

**Lemma 3.** *Let  $\epsilon_D$  and  $\delta$  denote the advantage of distinguisher  $\mathcal{D}_G$  and forger  $\mathcal{F}$  respectively, and let  $\alpha$  and **NoAbort** defined as before. Then*

$$\delta \leq n \cdot (\alpha \epsilon_D + \Pr[\text{NoAbort}]) .$$

The following notation will be useful in the proof. For any  $v \in V$ , let  $W(v) = V_G \cap \text{Pred}(v) \setminus \{v\}$  denote the set of all expansion vertices which are predecessors of  $v$ . Also, given a vertex  $v \in V$  and a vertex  $u \in W(v)$ , let  $\text{Pred}_v(\leq$

$u) = \bigcup_{u' \leq u, u' \in W(v)} \text{Pred}(u')$  the cut formed by the union over  $u' \leq u$  of all sets  $\text{Pred}(u') \subset \text{Pred}(v)$ . (Recall that  $\leq$  is a total order relation over  $V_G$ ) Also, let  $\{\sigma: \text{Pred}_u(\leq v)\}$  denote the set of all labeled cuts on  $\text{Pred}_v(\leq u)$ ; as before, for  $y_1, y_2 \in R^2$ , let  $\sigma: \text{Pred}_u(\leq v)_{y_1, y_2}$  denote the set of all labeled cuts compatible with  $(u, y_1, y_2)$ . (We stress that the compatibility is with respect to vertex  $u$ , that is,  $\sigma(e_1); \sigma(e_2) = y_1, y_2$ ). As before, if  $x \in R$  is uniformly distributed, the set  $\{\sigma: \text{Pred}_v(\leq u)_{G(x)}\}$  is denoted by  $\{\sigma: \text{Pred}_v(\leq u)_{G(\cdot)}\}$ . Notice that in this extended definition,  $\text{Pred}_v(\leq u) \subset \text{Pred}(v)$  and therefore a labeled cut for  $\text{Pred}(v)$  can be computed from any labeled cut in  $\{\sigma: \text{Pred}_u(\leq v)\}$ .

*Proof (Lemma 3).* By definition,  $\epsilon_D = p_1 - p_0$ , where  $p_1$  and  $p_0$  denote the probability that  $\mathcal{D}_G(y_1, y_2) = 1$  when  $x \xleftarrow{R} R, y_1, y_2 \leftarrow G(x)$  and the probability that  $\mathcal{D}_G(y_1, y_2) = 1$  when  $y_1, y_2 \xleftarrow{R} R^2$ , respectively. Consider the following two experiments, which we denote **Exp**<sub>1</sub> and **Exp**<sub>0</sub>. In the first one, we choose  $H \in \mathcal{H}$  uniformly at random,  $v \in V_H \cup V_G$  with probability proportional to  $|W(v)|$ ,  $u \in W(v)$  and  $\sigma_u \in \{\sigma: \text{Pred}_v(\leq u)_{G(\cdot)}\}$  uniformly at random; we then compute  $\sigma_v$  as an extension of  $\sigma_u$  by  $\sigma_v = \Pi_{\text{Pred}(v)}(\sigma_u)$  and finally call  $\mathcal{A}$  on input  $(H, v, \sigma_v)$ . The second experiment, **Exp**<sub>0</sub>, is similar to the previous one, with the exception that  $\sigma_u$  is drawn at random from  $\{\text{Pred}_v(\leq u)_{y_1, y_2}\}$  for  $y_1, y_2 \xleftarrow{R} R^2$ . Let  $q_1(v', u')$  and  $q_0(v', u')$  denote the probability procedure  $\mathcal{A}$  does not abort in **Exp**<sub>1</sub> and **Exp**<sub>0</sub> respectively, conditioned on the event that  $v = v'$  and  $u = u'$  are chosen in each experiment.

Let  $\alpha = \frac{1}{n} \sum_{v \in V_H \cup V_G} |W(v)|$  be the average number of expansion vertices of a random vertex in the graph. We claim that,

$$p_1 = \frac{1}{n\alpha} \cdot \sum_{v \in V_H \cup V_G} \sum_{u \in W(v)} q_1(v, u) \quad \text{and} \quad p_0 = \frac{1}{n\alpha} \cdot \sum_{v \in V_H \cup V_G} \sum_{u \in W(v)} q_0(v, u) \quad (1)$$

and that for all  $v \in V_H \cup V_G$ ,  $u \in W(v) \cup \{v\}$

$$q_0(v, u^*) = q_1(v, u) \quad (2)$$

$$\sum_{v \in V_H \cup V_G} q_1(v, \bar{v}) \geq \delta \quad (3)$$

$$\sum_{v \in V_H \cup V_G} q_0(v, v^*) = n \cdot \Pr[\text{NoAbort}] \quad (4)$$

where  $w^* = \max_{w' < w} (w')$ , denotes the biggest vertex in  $V_G$  smaller than  $w \in V_G$  and  $\bar{v} = \min_{v \in V_G} (v)$  is the “smallest” expansion vertex in  $V_G$  (where “biggest” and “smallest” refer to the  $\leq$  ordering relation).

Before proving these claims, we use them to finish the proof of the lemma. Using equations (1)-(4), we have

$$\epsilon_D = \frac{1}{n\alpha} \sum_{v \in V_H \cup V_G} \{q_1(v, \bar{v}) - q_0(v, v^*)\} \geq \frac{1}{n\alpha} \cdot (\delta - n \cdot \Pr[\text{NoAbort}])$$

which gives the desired result.

We now justify the claimed equations (14) by analyzing each them in turn. To justify the first part of (11), notice that by definition of  $p_1$  and standard conditioning we have

$$\begin{aligned} p_1 &= \Pr \left[ \mathcal{A}(H, v, \sigma_v) \neq \text{abort} : v \stackrel{W}{\leftarrow} V_G \cup V_H, u \stackrel{R}{\leftarrow} W(v), H \stackrel{R}{\leftarrow} \mathcal{H}, x \stackrel{R}{\leftarrow} R, \right. \\ &\quad \left. \sigma_u \stackrel{R}{\leftarrow} \text{Pred}_v(\leq u)_{G(x)} \right] = \sum_{v \in V_H \cup V_G} \sum_{u \in W(v)} q_1(v, u) \cdot \Pr[u \mid v] \cdot \Pr[v] \\ &= \sum_{v \in V_H \cup V_G} \frac{\Pr[v]}{|W(v)|} \sum_{u \in W(v)} q_1(v, u) \end{aligned}$$

where  $v \stackrel{W}{\leftarrow} V_G \cup V_H$  means vertex  $v$  is drawn from set  $V_G \cup V_H$  with probability proportional to  $|W(v)|$ . Since, for all  $v \in V$ ,  $\Pr[v] = |W(v)|/(n\alpha)$  Equation (11) holds. The second part of (11) follows from a similar argument.

We justify Equation (2) as follows. Fix  $v \in V_H \cup V_G$  and  $u \in W(v) \cup \{v\}$ . Consider experiment  $\mathbf{Exp}_0$ , and assume  $v$  and  $u^* \in W(v)$  are the vertices chosen. First of all, notice that  $\text{Pred}_v(\leq u^*) \subset \text{Pred}_v(\leq u)$  because  $u^* \leq u$ , and thus,  $\sigma_u$  can be computed from  $\sigma_{u^*}$ . Second, assume  $v \in V_G$ . Since the labeled cut  $\sigma_{u^*} \in \sigma : \text{Pred}_v(\leq u^*)$  is chosen uniformly at random, there is no other expansion node in any path from  $u' \leq u^*$  and  $u$ , and  $H$  is regular, the induced labeled cut  $\sigma_u = \Pi_{\text{Pred}(u)}(\sigma_{u^*}) \in \sigma : \text{Pred}_v(\leq u)$  is such that  $\sigma_u(e_1); \sigma_u(e_2) = G(x)$  for some  $x \in R$  uniformly distributed ( $e_1$  and  $e_2$  are edges leaving vertex  $u$ ). The same argument when  $v \in V_H$  boils down to  $\sigma_u(e) = H(x_1; x_2)$  for uniformly distributed  $x_1; x_2 \in R^2$  and  $e$  the only leaving edge of  $u$ . Thus,  $\sigma_u \in \sigma : \text{Pred}_v(\leq u)_R$ , and  $q_0(v, u^*) = q_1(v, u)$ .

To justify Equation (3) we notice that when distinguisher  $\mathcal{D}_G$  chooses  $u = \bar{v}$ , the distribution of the public key and signature so computed by  $\mathcal{A}$  from  $\sigma_u$  follows the same distribution than the forger expects in the one-chosen-message attack and, thus, the output of the forger is independent of the choice of  $v$ .

$$\begin{aligned} \sum_{v \in V_H \cup V_G} q_1(v, \bar{v}) &= \\ \sum_{v \in V_H \cup V_G} \Pr[\mathcal{F}(m, \sigma_m) = (m', \sigma'), m \neq m', v \in \mu(m), v \notin \mu(m')] &\geq \delta \end{aligned}$$

where the last inequality follows from that, for any  $m, m' \in \mathcal{M}$ , if  $m \neq m'$  there always exists  $v \in V_H \cup V_G$  such that  $v \in \mu(m)$  but  $v \notin \mu(m')$ , otherwise  $m$  and  $m'$  would be comparable.

It remains to prove Equation (4). This follows from  $q_0(v, v^*) = q_1(v, v) = \Pr[\text{NoAbort} \mid v]$  and from vertex  $v \in V_H \cup V_G$  being chosen uniformly at random in the experiment that defines the event  $\text{NoAbort}$ . This concludes the proof of the lemma.

*Proof (Theorem 7).* Immediate from Lemma 2 and Lemma 3.

## 6 Extensions

In this section we consider extensions of the basic security results presented in the previous sections. The first one concerns relaxing the security assumptions about the underlying primitives  $G, H$ . The second applies the ideas in our proof of security to build provably secure signature schemes with special algebraic properties.

**UNIVERSAL ONE-WAY HASH FUNCTIONS:** The collision-resistance requirement on the hash function family  $\mathcal{H}$  can be relaxed to *universal one-wayness* as defined by Naor and Yung [NY89]. Recall that universal one-way hash function (UOWHF) families are such that it is hard to find a colliding pair  $x \neq x'$  such that  $H(x) = H(x')$  but the adversary must select  $x$  before  $H$  is given to it. We modify our GBOTS construction, so that for each compression vertex  $v$  a different randomly chosen function  $H_v \in \mathcal{H}$  is used. The security argument in this case is modified as follows. In order to compute  $\sigma_\top = \Pi_{\{v_\top\}}(\sigma_v)$ , algorithm  $\mathcal{A}(H, v, \sigma)$  picks a hash function  $H_v \in \mathcal{H}$  uniformly at random anew to compute the label of each edge leaving a compression vertex with the exception of the edge corresponding to  $v$ , for which  $H$  is used. Thus, adversary  $\mathcal{I}_H$  needs only to pick ahead a random value  $x \in R^2$  and, once given a target hash function  $H$ , to use procedure  $\mathcal{A}$  to invert  $H(x)$ . Similarly, for  $\mathcal{C}_H$  it suffices to guess the compression vertex where the collision given by Lemma 4 will be found, and use the target hash function  $H$  there. Adversaries  $\mathcal{I}_H$  and  $\mathcal{C}_H$  remain the same. The remaining security argument does not differ substantially from the one presented in Section 5. We point out that regular universal one-way hash functions and one-to-one pseudorandom generators can be constructed from any one-way permutation [NY89, CMR98].

**MAPPING MESSAGES TO EDGES (OR VERTICES):** In this paper, we associate values to edges in the graph and functions to vertices. This approach can be seen as dual to the one used in [BM94], which associates values to vertices and function to edges. Both approaches are essentially equivalent from a syntax viewpoint and in terms of the class of schemes they yield. From a foundational viewpoint, we believe that the approach presented here is conceptually simpler.

**GRAPH BASED ALGEBRAIC SIGNATURE SCHEMES:** Algebraic signature schemes are signature schemes in which signatures for (certain) new messages can be produced by combining signatures with a restricted set of operations. Since these operations do not require knowledge of the secret key, algebraic signatures are not signature schemes in the standard interpretation of the term, but they are a new cryptographic primitive. They are useful in contexts where possession of signatures of certain messages automatically entitles possession of signatures of new messages, such as in credential systems. Credentials may be implemented as signed documents which specify capabilities (or attributions) to be granted to the credential holder. Thus, if implemented with the appropriate algebraic signature, the possession of one or more credentials (signatures) will automatically

enable the computation of the entitled credentials without the involvement of the original signer. Algebraic signatures were originally suggested by Rivest [MR02].

Informally, an algebraic signature scheme consists of three algorithms  $\mathcal{AS} = (\text{KG}, \text{Sig}, \text{Vf})$  and a two set of operations  $\mathcal{O} = \{f_1, f_2, \dots, f_q\}$  and  $\mathcal{S} = \{g_1, g_2, \dots, g_s\}$ , where each  $f_i$  (resp.  $g_i$ ) is a function that takes one or more messages (resp. signatures) as inputs and produces one message (resp. signature) as output.  $\text{KG}$ ,  $\text{Sig}$ , and  $\text{Vf}$  are as in any digital signature scheme (see Section 2.1). We require that if  $\delta_1, \dots, \delta_t$  are valid signatures for  $m_1, \dots, m_t$  then  $g_i(\delta_1, \dots, \delta_t)$  is a valid signature for  $f_i(m_1, \dots, m_t)$  for all appropriate  $f_i, g_i$ . Notice that signatures so generated are subject to existential forgery under chosen message attacks, so a new definition of security is required. Let  $\text{span}(\mathcal{O}, \{m_1, \dots, m_t\})$  be the set of all messages computable from  $\{m_1, \dots, m_t\}$  by applying functions in  $\mathcal{O}$  on them. The security of algebraic signatures is defined in terms of unforgeability against chosen-message attacks, where by convention, the forger is deemed successful only if it outputs a signature of a message  $m$  not in the set  $\text{span}(\mathcal{O}, \{m_1, \dots, m_t\})$ .

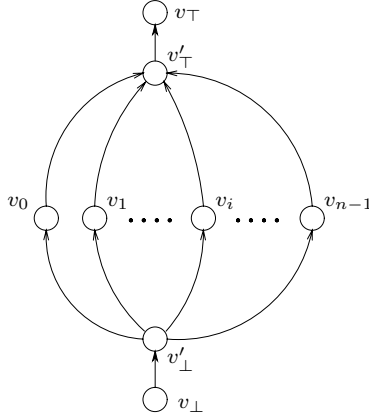
Graph-based one-time signatures can be used to build very efficient algebraic signatures. Indeed, for practical functions  $f_i$ , it is possible to build graphs such that  $f_i$  is embedded in the order relation  $\sqsubseteq$ . That is, if  $f_i(m_1, m_2) = m_3$ , then there exists a labeling  $\sigma: \mu(m_3)$  which can be computed from labellings  $\sigma_1: \mu(m_1)$  and  $\sigma_2: \mu(m_2)$  and it is consistent with them.

Notice that the proof of security of Section 4 and Section 5 can be easily modified to prove that our (graph-based) algebraic signature scheme  $\mathcal{AS}$  is secure. Indeed, the only technical difference is that the forger  $\mathcal{F}$  can request multiple signatures  $\sigma_m: \mu(m)$ . This can be easily factored in by modifying Procedure  $\mathcal{A}$  so each signature  $\sigma_m$  is computed from  $\sigma_v$  (or  $\mathcal{A}$  aborts, if not possible). Since the forger  $\mathcal{F}$  must output  $(m', \sigma')$  for  $m'$  not in  $\text{span}(\mathcal{O}, \cup_m \mu(m))$ , there must exist  $v \in \cup_m \mu(m)$  so  $v \notin \mu(m')$  and the argument goes through. The rest of the proof is identical and, in particular, adversaries  $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$  remain the same, given black-box access to  $\mathcal{A}$ .

**CONCRETE CONSTRUCTIONS OF ALGEBRAIC SIGNATURE SCHEMES:** In this section we sketch concrete graph constructions that yield algebraic signature schemes with respect to (a) union and subset operations, and (b) union and super-intersection operations. (Recall that the super-intersection of sets  $A$  and  $B$ , denoted  $A \odot B$ , is the collection of all sets  $S$  such that  $A \cap B \subseteq S \subseteq A \cup B$ .)

Let  $\mathcal{M}$  be the set of all subset of  $n$  elements, where we denote such elements as  $t_0, \dots, t_{n-1}$ . Consider the graph shown in Figure 6. (Although the figure shows vertices  $v_i$  having indegree and outdegree 1, and the vertices  $v'_\perp$  and  $v'_\top$  having outdegree and indegree  $n$ , respectively, it is easy to cast this graph as one with the properties considered in this work. Indeed, it suffices to replace each vertex  $v_i$  with a small subgraph of 2 compression and 2 expansion vertices, and to connect each  $v_i$  to both  $v'_\perp$  and  $v'_\top$  by simple tree construction).

We map every set  $S$  into the set of vertices  $\mu(S) = C$  defined as follows: vertices  $v_\perp$  and  $v'_\perp$  are in  $C$ , and vertices  $v_i$  are in  $C$  if and only if  $t_i \notin S$ . Notice  $C$  is a valid cut for any set  $S$ . Given a labeled cut  $\sigma: \mu(S)$  the labeling for any  $C' = \mu(S')$  such that  $S' \subseteq S$  can be computed by projecting  $\sigma: \mu(S)$  on  $C'$ .



**Fig. 6.** DAG for algebraic scheme with operations  $\{\cup, \text{subset}\}$ .

The union operation is defined similarly, since given labeled cuts  $\sigma_1: \mu(S_1)$  and  $\sigma_2: \mu(S_2)$  a consistent labeled cut for  $\mu(S_1 \cup S_2)$  can be computed.

A algebraic signature scheme for the  $\{\cup, \odot\}$  operations can be build by using two graphs  $\mathcal{G}_1$  and  $\mathcal{G}_2$  each one like the one described above. In this case, given a set  $S$ , we define the cut on the first graph by using the above shown rule, while for the second case we “invert” the condition, and we include the corresponding vertices only if  $t_i \in S$ . It is an easy exercise to verify that such mapping allows the computation of labeled cuts corresponding to the union and super-intersection of two sets  $S_1$  and  $S_2$ , given labeled cuts  $\sigma_1: \mu(S_1)$  and  $\sigma_2: \mu(S_2)$ .

## 7 Conclusions

In this paper, we analyze graph based signatures from a security viewpoint and give sufficient conditions, namely the existence of one-way permutations, under which the signature scheme is secure in the standard complexity model (no random oracles). Additionally, we present a security proof which uses a new hybrid argument where the number of hybrid distributions may be exponential. We believe this technique is of independent interest. We also propose a new paradigm for the construction of algebraic signature schemes, which are new useful primitives for applications where controlled “forgeability” of signatures is needed, as in credential systems.

## Acknowledgments

We want to thank Bogdan Warinschi, and Yee Hea Ann for helpful comments and discussions.

## References

- AR00. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In ASIACRYPT'2000, LNCS 1976, pages 116–129. Springer-Verlag, 2000.
- BC93. J. N. E. Bos and D. Chaum. Provable unforgeable signatures. In CRYPTO'92, LNCS 740, pages 1–14. Springer-Verlag, 1993.
- BKR94. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In CRYPTO'94, LNCS 839. Springer-Verlag, 1994.
- BM84. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. In *Siam Journal of Computing*, 13(4):850–864, 1984.
- BM92. M. Bellare and S. Micali. How to sign given any trapdoor function. In *Journal of Cryptology*, 39(1):214–233, 1992.
- BM94. D. Bleichenbacher and U. M. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In CRYPTO'94, LNCS 839, pages 75–82. Springer-Verlag, 1994.
- BM96a. D. Bleichenbacher and U. M. Maurer. On the efficiency of one-time digital signatures. In ASIACRYPT'96, LNCS 1163, pages 145–158. Springer-Verlag, 1996.
- BM96b. D. Bleichenbacher and U. M. Maurer. Optimal tree-based one-time digital signature schemes. In STACS'96, LNCS 1046, pages 363–374. Springer-Verlag, 1996.
- BM99. M. Bellare and S. Miner. A forward-secure digital signature scheme. In CRYPTO'99, LNCS 1666, pages 431–448. Springer-Verlag, 1999.
- BN02. M. Bellare and G. Neven. Transitive Signatures based on Factoring and RSA. In ASIA-CRYPT'02, (these proceedings).
- BR97. M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In CRYPTO'97, LNCS 1294, pages 470–484. Springer-Verlag, 1997.
- CLR92. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. In *Introduction to algorithms*. MIT Press and McGraw-Hill Book Company, 6th ed., 1992.
- CMR98. R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. STOC'98, pages 131–140. ACM, 1998.
- DN94. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. In CRYPTO'94, LNCS 839, pages 234–246. Springer-Verlag, 1994.
- EGM96. S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Journal of Cryptology*, 9(1):35–67, 1996.
- GMR88. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. In *Siam Journal of Computing*, 17(2):281–308, 1988.
- HPT97. R. Hauser, A. Przygienda, and G. Tsudik. Reducing the cost of security in link state routing. In *Symposium on Network and Distributed Systems Security (NDSS '97)*, pages 93–99, Internet Society, 1997.
- HM02. A. Hevia and D. Micciancio. The provable security of Graph-Based One-Time Signatures and extensions to algebraic signature schemes. Full version of this paper, available via <http://www-cse.ucsd.edu/users/ahevia>.
- JMSW02. R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In CT-RSA '2002, LNCS 2271, pages 244–262. Springer-Verlag, 2002.

- Lam79. L. Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International, 1979.
- Mer82. R. C. Merkle. In *Secrecy, Authentication, and Public Key Systems*, vol. 18 of *Computer science. Systems programming*. UMI Research Press, 1982.
- Mer87. R. C. Merkle. A digital signature based on a conventional encryption function. In CRYPTO'87, LNCS 293, pages 369–378. Springer-Verlag, 1987.
- Mer90. R. C. Merkle. A digital signature based on a conventional encryption function. In CRYPTO'89, LNCS 435, pages 428–446. Springer-Verlag, 1990.
- MM82. C. H. Meyer and S. M. Matyas. In *Cryptography: A New Dimension in Computer Data Security*. John Wiley and Sons, New York, 1982.
- MMM02. T. Malkin, D. Micciancio, and S. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In EURO-CRYPT'2002, LNCS 2332, pages 400–417. Springer-Verlag, 2002.
- MR02. S. Micali and R. L. Rivest. Transitive signature schemes. In CT-RSA '2002, LNCS 2271, pages 236–243. Springer-Verlag, 2002.
- NY89. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. STOC'89, pages 33–43. ACM, 1989.
- Per01. A. Perrig. The BiBa one-time signature scheme and broadcast authentication protocol. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 28–37. ACM, 2001.
- Rab78. M. O. Rabin. Digitalized signatures. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.
- Roh99. P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 93–100, ACM, 1999.
- RSA78. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public-key cryptosystems. In *Communications of the ACM*, 21(2):120–126, 1978.
- Sch90. C. Schnorr. Efficient identification and signatures for smartcards. In CRYPTO'89, LNCS 435, pages 239–252. Springer-Verlag, 1990.
- Vau92. S. Vaudenay. One-time identification with low memory. In *Eurocode 92*, CISM Courses and Lectures, no. 339, pages 217–228, Springer-Verlag, 1992.
- Yao82. A. Yao. Theory and applications of trapdoor functions. FOCS'82, pages 80–91. IEEE, 1982.



# Transitive Signatures Based on Factoring and RSA

Mihir Bellare<sup>1</sup> and Gregory Neven<sup>2</sup>

<sup>1</sup> Dept. of Computer Science & Engineering, University of California, San Diego,  
9500 Gilman Drive, La Jolla, CA 92093, USA,

[mihir@cs.ucsd.edu](mailto:mihir@cs.ucsd.edu), <http://www-cse.ucsd.edu/users/mihir>

<sup>2</sup> Dept. of Computer Science, Katholieke Universiteit Leuven,  
Celestijnenlaan 200A, B-3001 Heverlee, Belgium,

[Gregory.Neven@cs.kuleuven.ac.be](mailto:Gregory.Neven@cs.kuleuven.ac.be), <http://www.cs.kuleuven.ac.be/~gregory>

**Abstract.** We present novel realizations of the transitive signature primitive introduced by Micali and Rivest [12], and also provide an answer to an open question they raise regarding the security of their RSA based scheme. Our schemes provide performance improvements over the scheme of [12].

**Keywords:** Signatures, transitive signatures, RSA.

## 1 Introduction

**THE CONCEPT.** The context envisioned by Micali and Rivest [12] is that of dynamically building an authenticated graph, edge by edge. The signer, having secret key  $tsk$  and public key  $tpk$ , can at any time pick a pair  $i, j$  of nodes and create a signature of  $\{i, j\}$ , thereby adding edge  $\{i, j\}$  to the graph. A composability property is required: given a signature of an edge  $\{i, j\}$  and a signature of an edge  $\{j, k\}$ , anyone in possession of the public key can create a signature of the edge  $\{i, k\}$ . Security asks that this limited class of forgeries be the only possible ones. (I.e., without  $tsk$ , it should be hard to create a valid signature of edge  $\{i, j\}$  unless  $i, j$  are connected by a path whose edges have been explicitly authenticated by the signer.) Thus the authenticated graph at any point is the transitive closure of the graph formed by the edges explicitly authenticated by the signer, whence the name of the concept. We refer the reader to Section 2 for formal definitions and to [12] for motivation and potential applications.

**REALIZING THE CONCEPT.** A transitive signature scheme can be trivially realized by accepting, as a valid signature of  $\{i, j\}$ , any chain of signatures that authenticates a sequence of edges forming a path from  $i$  to  $j$ . Two issues lead [12] to exclude this: the growth in signature size, and the loss of privacy incurred by having signatures carry information about their history. The main result of [12] is a (non-trivial) transitive signature scheme (we call it the MRTS scheme) proven to be (transitively) unforgeable under adaptive chosen-message attack (see Section 2 for formal definitions) assuming that the discrete logarithm problem is hard in an underlying prime-order group and assuming security of an

underlying standard signature scheme. They also present a natural RSA based transitive signature scheme but point out that even though it seems secure, and a proof of unforgeability under non-adaptive chosen-message attacks exists, there is no known proof of unforgeability under *adaptive* chosen-message attacks. They thereby highlight the fact that in this domain, adaptive attacks might be harder to provably protect against than non-adaptive ones.

In summary, transitive signatures (unforgeable under adaptive chosen-message attacks) at this point have just a single realization, namely the MRTS scheme. It is standard practice in cryptography to seek new and alternative realizations of primitives of potential interest, both to provide firmer theoretical foundations for the existence of the primitive by basing it on alternative conjectured hard problems and to obtain performance improvements. This paper presents new schemes that accomplish both of these objectives, and also provides an answer to the question about the RSA scheme.

**THE NODE CERTIFICATION PARADIGM.** It is worth outlining the node certification based paradigm introduced by the MRTS scheme, which will be our starting point. The signer's keys include those of a standard digital signature scheme, and the public key includes additional items. (In the MRTS scheme, this is a group  $\mathbb{G}$  of prime order  $q$  and a pair of generators of  $\mathbb{G}$ .) The signer associates to each node  $i$  in the current graph a *node certificate* consisting of a *public label*  $L(i)$  and a signature of  $(i, L(i))$  under the standard scheme. The signature of an edge contains the certificates of its endpoints plus an *edge label*  $\delta$ . Verification of the signature of an edge involves relating the edge label to the public labels of its endpoints as provided in the node certificates and verifying the standard signatures in the node certificates. Composition involves algebraic manipulation of edge labels.<sup>1</sup>

The paradigm is useful, but brings an associated cost. Producing a signature for an edge can involve computing two normal signatures. The length of an edge signature, containing two node certificates each including a standard signature, can be large even if the edge labels are small.

### 1.1 Transitive Signatures Based on Factoring

Our first factoring-based transitive signature (FBTS-1) scheme stays within the node certification paradigm but, by implementing label algebra via square-roots modulo a composite, provides security based on factoring while reducing some costs compared to MRTS.

**FBTS-1.** The signer has keys for a standard signature scheme, and its public key additionally includes a modulus  $N$  product of two large primes. The public label of a node  $i$  is a quadratic residue  $L(i) \in \mathbb{Z}_N^*$ , and an edge label of edge  $\{i, j\}$  is a square root of  $L(i)L(j)^{-1} \bmod N$  assuming  $i < j$ . Composition involves multiplying edge labels modulo  $N$ . We prove that FBTS-1 is unforgeable

<sup>1</sup> Note that the signer is stateful, and once quantities associated to a node are created, they are stored and re-used for all edges adjacent to this node. This is important for security. See Section 3 for a discussion of how state can be eliminated.

under adaptive chosen-message attack, assuming the hardness of factoring the underlying modulus, and assuming security of the underlying standard signature scheme. The delicate part of this proof is an information-theoretic lemma showing that, even under an adaptive chosen-message attack, for any  $\{i, j\}$  not in the transitive closure of the current graph, an adversary has zero advantage in determining which of the square roots of  $L(i)L(j)^{-1}$  is held by the signer.

With regard to costs, we are interested in the computational cost of signing an edge (in the worst case that both endpoints of the edge are not in the current graph); the computational cost of verifying a candidate signature of an edge; the computational cost of composing two edge signatures to obtain another; and the size of a signature. Since FBTS-1 continues to employ the node certification paradigm, it incurs the same costs as MRTS from the use of the standard signature scheme. However, as Figure 1 indicates, it is otherwise cheaper than MRTS for signing and verifying, reducing the extra cost from cubic (exponentiation) to quadratic (a couple of multiplications and an inverse).

**FBTS-2: ELIMINATING NODE CERTIFICATES.** FBTS-1 is amenable to a modification which eliminates the need for node certificates and thereby removes the standard signature scheme, and all its associated costs, from the picture. The signer’s public key is a modulus  $N$  product of two primes  $p, q$  that make up the signer’s secret key. The public label of a node  $i$  is not chosen by the signer but rather specified via the output of a public hash function applied to  $i$ . (A difficulty, addressed in Section 4, is that the hash output might not be a quadratic residue.) We prove that FBTS-2 is unforgeable under adaptive chosen-message attack, assuming the hardness of factoring the underlying modulus, in a model where the hash function is a random oracle [5].

Scheme	Signing cost	Verification cost	Composition cost	Signature size
MRTS	2 stand. sigs 2 exp. in $\mathbb{G}$	2 stand. verifs 1 exp. in $\mathbb{G}$	2 adds in $\mathbb{Z}_q$	2 stand. sigs 2 points in $\mathbb{G}$ 2 points in $\mathbb{Z}_q$
FBTS-1	2 stand. sigs $O( N ^2)$ ops	2 stand. verifs $O( N ^2)$ ops	$O( N ^2)$ ops	2 stand. sigs 3 points in $\mathbb{Z}_N^*$
FBTS-2	4 sq. roots in $\mathbb{Z}_N^*$	$O( N ^2)$ ops	$O( N ^2)$ ops	1 point in $\mathbb{Z}_N^*$
RSATS-1	2 stand. sigs 2 RSA encs	2 stand. verifs 1 RSA enc.	$O( N ^2)$ ops	2 stand. sigs 3 points in $\mathbb{Z}_N^*$
RSATS-2	1 RSA dec.	1 RSA enc.	$O( N ^2)$ ops	1 point in $\mathbb{Z}_N^*$

**Fig. 1.** Cost comparisons amongst transitive signature schemes. The word “stand.” refers to operations of the underlying standard signature scheme, which are eliminated for FBTS-2 and RSATS-2.  $\mathbb{G}$  denotes the group of prime order  $q$  used in MRTS, and  $N$  denotes a modulus product of two primes as used in the other schemes. Abbreviations used are: “exp.” for an exponentiation in the group; “RSA enc.” for an RSA encryption; “RSA dec.” for an RSA decryption performed given the decryption exponent; “sq. root” for a square root modulo  $N$  performed using the prime factors of  $N$ ; and “ops” for the number of elementary bit operations in big-O notation.

As Figure 1 indicates, the major cost savings is elimination of all costs associated to the standard scheme. However, signing now requires computation of square roots modulo  $N$  by the signer based on the prime factorization of  $N$ , which has cost comparable to an exponentiation modulo  $N$ . Thus overall the main gain is the reduction in signature size.

This hash based modification is made possible by the fact that squaring modulo a composite is a trapdoor function. The MRTS scheme is not amenable to a similar hash-based modification since the discrete exponentiation function is not trapdoor over the prime order groups used in [12].

## 1.2 Transitive Signatures Based on RSA

**RSATS-1.** The RSA-based transitive signature scheme noted in [12] (that we call RSATS-1) employs the node certification paradigm. The signer has keys for a standard signature scheme. Its public key additionally includes an RSA modulus  $N$  and encryption exponent  $e$ , while its secret key includes the corresponding decryption exponent  $d$ . The public label of a node  $i$  is a point  $L(i) \in \mathbb{Z}_N^*$ , and the edge label of edge  $\{i, j\}$  is  $L(i)^d L(j)^{-d} \bmod N$  assuming  $i < j$ . Composition involves multiplying edge labels modulo  $N$ . One can prove that RSATS-1 is unforgeable under *non-adaptive* chosen-message attacks assuming the one-wayness of RSA and the security of the underlying standard signature scheme. No adaptive chosen-message attack that succeeds in forgery has been found, but neither has it been proven that RSATS-1 is unforgeable under adaptive chosen-message attack.

One might wonder why proofs exist for MRTS and FBTS-1 but remain elusive for RSATS-1 in spite of the obvious similarities between these schemes. The proofs for MRTS and FBTS-1 exploit the fact that there are multiple valid edge labels for any given edge in the graph, and that finding two different edge labels implies solving the underlying hard problem. With RSATS-1, the edge label is uniquely determined by the two node certificates, and this paradigm fails.

This situation (namely a scheme that appears to resist both attack and proof) is not uncommon in cryptography, and we suggest that it is a manifestation of the fact that the security of the scheme is relying on properties possessed by RSA but going beyond those captured by the assumption that RSA is one-way. Accordingly we seek an alternative, stronger assumption upon which a proof of security can be based.

We prove that RSATS-1 is unforgeable under adaptive chosen-message attacks under the assumption that RSA is secure under one-more-inversion (and the standard signature scheme is secure). This assumption was introduced by [2], who used it to prove the security of Chaum's blind signature scheme [7]. It was also used in [4] to prove security of the GQ identification scheme [10] against impersonation under active attack

Security under one more inversion considers an adversary given input an RSA public key  $N, e$ , and two oracles. The *challenge* oracle takes no inputs and returns a random target point in  $\mathbb{Z}_N^*$ , chosen anew each time the oracle is invoked.

Scheme	Proven to be unforgeable under <i>adaptive</i> chosen-message attack assuming	RO Model?
MRTS	Security of standard signature scheme Hardness of discrete logarithm problem in a group of prime order	No
FBTS-1	Security of standard signature scheme Hardness of factoring	No
FBTS-2	Hardness of factoring	Yes
RSATS-1	Security of standard signature scheme RSA is secure against one-more-inversion attack	No
RSATS-2	RSA is secure against one-more-inversion attack	Yes

**Fig. 2.** Provable security attributes of transitive signature schemes. We indicate the assumptions under which there is a proof of unforgeability under *adaptive* chosen-message attack, and whether or not the random oracle model is used.

The *inversion* oracle given  $y \in \mathbb{Z}_N^*$  returns  $y^d \bmod N$  where  $d$  is the decryption exponent corresponding to  $e$ . The assumption states that it is computationally infeasible for the adversary to output correct inverses of all the target points if the number of queries it makes to its inversion oracle is strictly less than the number of queries it makes to its challenge oracle. When the adversary makes one challenge query and no inversion queries, this reduces to the standard one-wayness assumption.

**RSATS-2.** The trapdooriness of the RSA function makes RSATS-1 amenable to the elimination of node certificates via hashing, based on ideas similar to the ones we introduced above. We present RSATS-2, a transitive signature scheme that is unforgeable under adaptive chosen-message attacks in the random oracle model assuming RSA is secure against one-more-inversion. The public label of a node  $i$  is not chosen by the signer but rather implicitly specified as the output of a hash function applied to  $i$ , and RSA decryption is used to compute edge labels. Finally we note that RSATS-2 is the only one of the schemes discussed here whose signer is naturally stateless.

Figures 1 and 2 summarize, respectively, the costs and provable-security attributes of the various schemes we have introduced, and compare them with the MRTS scheme.

### 1.3 Definitional Contributions

Regarding the composability property, Micali and Rivest [12, p. 238] (we have modified the notation to be consistent with ours) say: “... if someone sees Alice’s signatures on edges  $\{i, j\}$  and  $\{j, k\}$  then that someone can easily compute a valid signature on edge  $\{i, k\}$  that is indistinguishable from a signature on that edge that Alice would have produced herself.” This seems to suggest that composition is only required to work when the given signatures were explicitly produced by the signer, but in fact we want composition to work even if the given

signatures were themselves obtained via composition. Formulating an appropriate requirement turns out to be more delicate than one might imagine. One could require the simple condition that valid signatures (meaning, ones accepted by the verification algorithm relative to the signer’s public key) can be composed to yield valid signatures. (This would follow [11], who require a condition that implies this.) But this requirement is too strong in the current context. Indeed, the MRTS scheme does not meet it, meaning there are valid signatures which, when composed, yield an invalid signature. The same is true for our schemes.

It can be proved that for MRTS and our schemes, finding valid signature inputs that make the composition algorithm return an invalid signature is computationally hard assuming the scheme is secure. But we prefer to not tie correctness of composition to security. Instead, we formulate correctness of composition via a recursive requirement that says that as long as one obtains signatures either directly via the signer or by applying the composition operation to signatures previously legitimately obtained or generated, then the resulting signature is valid. (This would be relatively easy to formulate if the signer was stateless, but needs more care due to the fact that the natural formulation of transitive signature schemes often results in a stateful signer.) As part of the formalization we provide in Definition 1, we follow [11] and require a very strong form of the indistinguishability requirement mentioned in the quote above, namely that the signature output by the composition algorithm is not just indistinguishable from, but identical to, the one the signer would have produced. (As argued in [11], this guarantees privacy.) The MRTS scheme, as well as all our schemes, meet this strong definition.

## 1.4 Related Work

Transitive signatures are one case of a more general concept promulgated by Rivest [14] in talks, namely that of signature schemes that admit forgery of signatures derived by some specific operation on previous signatures but resist other forgeries. Johnson, Molnar, Song and Wagner [11] formalize a notion of homomorphic signature schemes that captures this. Context Extraction Signatures, as introduced earlier by [15], as well as redactable signatures and set-homomorphic signatures [11], fall in this framework. A signature scheme that is homomorphic with respect to the prefix operation is presented by Chari, Rabin and Rivest [6].

## 2 Definitions

NOTATION. We let  $\varepsilon$  denote the empty string and  $\parallel$  the concatenation operator on strings. We let  $\mathbb{N} = \{1, 2, \dots\}$  be the set of positive integers. The notation  $x \xleftarrow{R} S$  denotes that  $x$  is selected randomly from set  $S$ . If  $A$  is a possibly randomized algorithm then the notation  $x \xleftarrow{R} A(a_1, a_2, \dots)$  denotes that  $x$  is assigned the outcome of the experiment of running  $A$  on inputs  $a_1, a_2, \dots$ .

GRAPHS. All graphs in this paper are undirected. A graph  $G^* = (V^*, E^*)$  is said to be *transitively closed* if for all nodes  $i, j, k \in V^*$  such that  $\{i, j\} \in E^*$  and

$\{j, k\} \in E^*$ , it also holds that  $\{i, k\} \in E^*$ ; or in other words, edge  $\{i, j\} \in E^*$  whenever there is a path from  $i$  to  $j$  in  $G^*$ . If  $G = (V, E)$  is a graph, its *transitive closure* is the graph  $\tilde{G} = (V, \tilde{E})$  where  $\{i, j\} \in \tilde{E}$  iff there is a path from  $i$  to  $j$  in  $G$ . Note that the transitive closure of any graph  $G$  is a transitively closed graph. Also note that any transitively closed graph can be partitioned into connected components such that each component is a complete graph.

**TRANSITIVE SIGNATURE SCHEMES AND THEIR CORRECTNESS.** A *transitive signature scheme*  $\text{TS} = (\text{TKG}, \text{TSign}, \text{TVf}, \text{Comp})$  is specified by four polynomial-time algorithms, and the functionality is as follows:

- The randomized *key generation* algorithm **TKG** takes input  $1^k$ , where  $k \in \mathbb{N}$  is the security parameter, and returns a pair  $(tpk, ts_k)$  consisting of a public key and matching secret key.
- The *signing algorithm* **TSign**, which could be stateful or randomized (or both), takes input the secret key  $ts_k$  and nodes  $i, j \in \mathbb{N}$ , and returns a value called an *original signature* of edge  $\{i, j\}$  relative to  $ts_k$ . If stateful, it maintains state which it updates upon each invocation.
- The deterministic *verification* algorithm **TVf**, given  $tpk$ , nodes  $i, j \in \mathbb{N}$ , and a candidate signature  $\sigma$ , returns either 1 or 0. In the former case we say that  $\sigma$  is a *valid signature* of edge  $\{i, j\}$  relative to  $tpk$ .
- The deterministic *composition* algorithm **Comp** takes  $tpk$ , nodes  $i, j, k \in \mathbb{N}$  and values  $\sigma_1, \sigma_2$  to return either a value  $\sigma$  or a symbol  $\perp$  to indicate failure.

The above formulation makes the simplifying assumption that the nodes of the graph are positive integers. In practice it is desirable to allow users to name nodes via whatever identifiers they choose, but these names can always be encoded as integers, so we keep the formulation simple.

Naturally, it is required that if  $\sigma$  is an original signature of edge  $\{i, j\}$  relative to  $ts_k$  then it is a valid signature of  $\{i, j\}$  relative to  $tpk$ .

As discussed in Section 1.3, formulating a correctness requirement for the composition algorithm is more delicate. Micali and Rivest [12] seem to suggest that composition is only required to work when the given signatures were explicitly produced by the signer, but in fact we want composition to work even if the given signatures were themselves obtained via composition. Furthermore the indistinguishability requirement is not formalized in [12].

Definitions taking these issues into account are however provided in [11]. They ask that whenever the composition algorithm is invoked on valid signatures (valid means accepted by the verification algorithm relative to the signer's public key) it returns the same signature as the signer would produce. This captures indistinguishability in a strong way that guarantees privacy. However, one implication of their definition is that whenever the composition algorithm is invoked on valid signatures, it returns a valid signature, and this last property is not true of known node certification based transitive signature schemes such as MRTS, FBTS-1 and RSATS-1. For these schemes, it is possible to construct examples of valid signature inputs that, when provided to the composition algorithm, result in the latter failing (returning  $\perp$  because it cannot compose) or

returning an invalid signature. (Roughly, this is because composition of a signature  $\sigma_1$  of  $\{i, j\}$  with a signature  $\sigma_2$  of  $\{j, k\}$  in these schemes requires that the public labels of node  $j$  as specified in  $\sigma_1$  and  $\sigma_2$  be the same. Validity of the individual signatures cannot guarantee this.)

This is not a weakness in the schemes, because in practice composition is applied not to arbitrary valid signatures but to ones that are legitimate, the latter being recursively defined: a signature is legitimate if it is either obtained directly by the signer, or obtained by applying the composition algorithm to legitimate signatures. What it points to is that we need to formulate a new correctness definition for composition that captures this intuition and results in a notion met by the known transitive signature schemes. Roughly, we would like a formulation that says that if the composition algorithm is invoked on legitimate signatures, then it returns the same signature as the signer would have produced. (Here, we are continuing to follow [11] in capturing indistinguishability by the strong requirement that composed signatures are identical to original ones, but weakening their requirement by asking that this be true not for all valid signature inputs to the composition algorithm, but only for legitimate inputs.)

The formalization would be relatively simple (the informal description above would pretty much be it) if the signing algorithm were stateless, but the natural formulation of numerous transitive signature schemes seems to be in terms of a stateful signing algorithm. In this case, it is not clear what it means that the output of the composition algorithm is the same as that of the signer, since the latter's output depends on its internal state which could be different at different times. To obtain a formal definition of correctness that takes into account the statefulness of the signing algorithm, we proceed as follows. We associate to any algorithm  $A$  (deterministic, halting, but not computationally limited) and security parameter  $k \in \mathbb{N}$  the experiment of Figure 3, which provides  $A$  with oracles

$$\text{TSign}(\text{tsk}, \cdot, \cdot) \text{ and } \text{Comp}(\text{tpk}, \cdot, \cdot, \cdot, \cdot),$$

where  $\text{tpk}$ ,  $\text{tsk}$  have been produced by running TKG on input  $1^k$ . In this experiment, the **TSign** oracle maintains state, and updates this state each time it is invoked. It also tosses coins anew at each invocation if it is randomized.

**Definition 1.** *We say that the transitive signature scheme **TS** is correct if for every (computationally unbounded) algorithm  $A$  and every  $k$ , the output of the experiment of Figure 3 is true with probability zero.*

The experiment computes a boolean **Legit** which is set to **false** if  $A$  ever makes an “illegitimate” query. It also computes a boolean **NotOK** which is set to **true** if a signature returned by the composition algorithm differs from the original one. To win,  $A$  must stay legitimate (meaning **Legit** = **true**) but violate correctness (meaning **NotOK** = **true**). The experiment returns **true** iff  $A$  wins. The definition requires that this happen with probability zero.

**SECURITY OF TRANSITIVE SIGNATURE SCHEMES.** We recall the notion of security of [12]. Associated to transitive signature scheme  $\text{TS} = (\text{TKG}, \text{TSign}, \text{TVf},$



```

 $(tpk, tsk) \xleftarrow{R} \text{TKG}(1^k)$ 
 $S \leftarrow \emptyset$ ;  $\text{Legit} \leftarrow \text{true}$ ;  $\text{NotOK} \leftarrow \text{false}$ 
Run  $A$  with its oracles until it halts, replying to its oracle queries as follows:
  If  $A$  makes TSign query  $i, j$  then
    If  $i = j$  then  $\text{Legit} \leftarrow \text{false}$ 
  Else
    Let  $\sigma$  be the output of the TSign oracle and let  $S \leftarrow S \cup \{(\{i, j\}, \sigma)\}$ 
    If  $\text{TVf}(tpk, i, j, \sigma) = 0$  then  $\text{NotOK} \leftarrow \text{true}$ 
  If  $A$  makes Comp query  $i, j, k, \sigma_1, \sigma_2$  then
    If  $[(\{i, j\}, \sigma_1) \notin S \text{ OR } (\{j, k\}, \sigma_2) \notin S \text{ OR } i, j, k \text{ are not all distinct}]$  then
       $\text{Legit} \leftarrow \text{false}$ 
    Else
      Let  $\sigma$  be the output of the Comp oracle and let  $S \leftarrow S \cup \{(\{i, k\}, \sigma)\}$ 
      Let  $\tau \leftarrow \text{TSign}(tsk, i, k)$ 
      If  $[(\sigma \neq \tau) \text{ or } \text{TVf}(tpk, i, k, \sigma) = 0]$  then  $\text{NotOK} \leftarrow \text{true}$ 
  When  $A$  halts, output  $(\text{Legit} \wedge \text{NotOK})$  and halt

```

**Fig. 3.** Experiment used to define correctness of the transitive signature scheme  $\text{TS} = (\text{TKG}, \text{TSign}, \text{TVf}, \text{Comp})$ .

$\text{Comp})$ , adversary  $F$  and security parameter  $k \in \mathbb{N}$  is an experiment, denoted

$$\mathbf{Exp}_{\text{TS}, F}^{\text{tu-cma}}(k),$$

that returns 1 if and only if  $F$  is successful in its attack on the scheme. The experiment begins by running **TKG** on input  $1^k$  to get keys  $(tpk, tsk)$ . If we are in the random oracle model, it also chooses the appropriate hash functions at random. It then runs  $F$ , providing this adversary with input  $tpk$  and oracle access to the function  $\text{TSign}(tsk, \cdot, \cdot)$ . The oracle is assumed to maintain state or toss coins as needed. Eventually,  $F$  will output  $i', j' \in \mathbb{N}$  and some value  $\sigma'$ . Let  $E$  be the set of all edges  $\{a, b\}$  such that  $F$  made oracle query  $a, b$ , and let  $V$  be the set of all integers  $a$  such that  $a$  is adjacent to some edge in  $E$ . We say that  $F$  *wins* if  $\sigma'$  is a valid signature of  $\{i', j'\}$  relative to  $tpk$  but edge  $\{i', j'\}$  is not in the transitive closure  $\tilde{G}$  of graph  $G = (V, E)$ . The experiment returns 1 if  $F$  wins and 0 otherwise. The *advantage* of  $F$  in its attack on  $\text{TS}$  is the function  $\mathbf{Adv}_{\text{TS}, F}^{\text{tu-cma}}(\cdot)$  defined for  $k \in \mathbb{N}$  by

$$\mathbf{Adv}_{\text{TS}, F}^{\text{tu-cma}}(k) = \Pr [\mathbf{Exp}_{\text{TS}, F}^{\text{tu-cma}}(k) = 1],$$

the probability being over all the random choices made in the experiment. We say that  $\text{TS}$  is *transitively unforgeable under adaptive chosen-message attack* if the function  $\mathbf{Adv}_{\text{TS}, F}^{\text{tu-cma}}(\cdot)$  is negligible for any adversary  $F$  whose running time is polynomial in the security parameter  $k$ .

**RO MODEL.** Some of our schemes will be defined in the random oracle model [5], which means that the algorithms **TSign**, **TVf**, **Comp** all have oracle access to one or more functions which in the correctness and security experiments are as-

sumed to be drawn at random from appropriate spaces. Formally, both the experiment of Figure 3 and  $\mathbf{Exp}_{\text{TS},F}^{\text{tu-cma}}(k)$  are augmented to choose a function mapping  $\{0,1\}^*$  to  $\{0,1\}^k$  at random, and the adversary, as well as the  $\text{TSig}$ ,  $\text{TVf}$ ,  $\text{Comp}$  algorithms, then get oracle access to this function. In Definition 1, the probability includes the choice of these functions, and so does the probability in the definition of  $\mathbf{Adv}_{\text{TS},F}^{\text{tu-cma}}(k)$ . Usually the scheme will need to construct out of the given function a function  $H$  with suitable range depending on the public key.

**STANDARD SIGNATURE SCHEMES.** Some of our schemes use an underlying standard digital signature scheme  $\text{SDS} = (\text{SKG}, \text{SSign}, \text{SVf})$ , described as usual via its polynomial-time key generation, signing and verification algorithms. We use the security definition of [9], where the forger  $B$  is given adaptive oracle access to the signing algorithm, and its advantage  $\mathbf{Adv}_{\text{SDS},B}^{\text{uf-cma}}(k)$  in breaking  $\text{SDS}$  is defined as the probability that it outputs a valid signature for a message that was not one of its previous oracle queries. The scheme  $\text{SDS}$  is said to be secure against forgery under adaptive chosen-message attack if  $\mathbf{Adv}_{\text{SDS},B}^{\text{uf-cma}}(\cdot)$  is negligible for every forger  $B$  with running time polynomial in the security parameter  $k$ .

### 3 A Transitive Signature Scheme Based on Factoring

**FACTORING PROBLEM.** A *modulus generator* is a randomized, polynomial-time algorithm that on input  $1^k$  returns a triple  $(N, p, q)$  where  $N = pq$ ,  $2^{k-1} \leq N < 2^k$ , and  $p, q$  are distinct, odd primes. There are numerous possible modulus generators which differ in the structure of the primes chosen or the distribution under which they are chosen. We do not restrict the type of generator, but only assume that the associated factoring problem is hard. Formally, for any modulus generator  $\text{MG}$ , adversary  $A$  and  $k \in \mathbb{N}$  we let

$$\mathbf{Adv}_{\text{MG},A}^{\text{fac}}(k) = \Pr \left[ r \in \{p, q\} : (N, p, q) \stackrel{R}{\leftarrow} \text{MG}(1^k) ; r \stackrel{R}{\leftarrow} A(k, N) \right] .$$

We say that *factoring is hard relative to*  $\text{MG}$  if the function  $\mathbf{Adv}_{\text{MG},A}^{\text{fac}}(\cdot)$  is negligible for every  $A$  whose running time is polynomial in  $k$ .

**THE SCHEME.** We are given a modulus generator  $\text{MG}$  and a standard digital signature scheme  $\text{SDS} = (\text{SKG}, \text{SSign}, \text{SVf})$ . We associate to them a transitive signature scheme  $\text{FBTS-1} = (\text{TKG}, \text{TSig}, \text{TVf}, \text{Comp})$  defined as follows:

- Given input  $1^k$ , the key generation algorithm  $\text{TKG}$  first runs  $\text{SKG}$  on input  $1^k$  to generate a key pair  $(\text{spk}, \text{ssk})$  for the standard signature scheme  $\text{SDS}$ . It then runs the modulus generator  $\text{MG}$  on input  $1^k$  to get a triple  $(N, p, q)$ . It outputs  $\text{tpk} = (N, \text{spk})$  as the public key of the transitive signature scheme and  $\text{tsk} = (N, \text{ssk})$  as the matching secret key. Note that the primes  $p, q$  are discarded and in particular not part of the secret key.
- The signing algorithm  $\text{TSig}$  maintains state  $(V, \ell, L, \Sigma)$  where  $V \subseteq \mathbb{N}$  is the set of all queried nodes, the function  $\ell: V \rightarrow \mathbb{Z}_N^*$  assigns to each node  $i \in V$  a *secret label*  $\ell(i) \in \mathbb{Z}_N^*$ , while the function  $L: V \rightarrow \mathbb{Z}_N^*$  assigns to each node  $i \in V$  a *public label*  $L(i)$ , and the function  $\Sigma: V \rightarrow \{0,1\}^*$  assigns to each

node  $i$  a standard signature on  $i\|L(i)$  under  $ssk$ . When invoked on inputs  $tsk, i, j$ , meaning when asked to produce a signature on edge  $\{i, j\}$ , it does the following:

If  $j < i$  then  $l \leftarrow j$ ;  $j \leftarrow i$ ;  $i \leftarrow l$  // swap  $i$  and  $j$  if necessary  
 If  $i \notin V$  then  $V \leftarrow V \cup \{i\}$ ;  $\ell(i) \xleftarrow{R} \mathbb{Z}_N^*$ ;  $L(i) \leftarrow \ell(i)^2 \bmod N$ ;  
 $\Sigma(i) \leftarrow \text{SSign}(ssk, i\|L(i))$   
 If  $j \notin V$  then  $V \leftarrow V \cup \{j\}$ ;  $\ell(j) \xleftarrow{R} \mathbb{Z}_N^*$ ;  $L(j) \leftarrow \ell(j)^2 \bmod N$ ;  
 $\Sigma(j) \leftarrow \text{SSign}(ssk, j\|L(j))$   
 $\delta \leftarrow \ell(i)\ell(j)^{-1} \bmod N$ ;  $C_i \leftarrow (i, L(i), \Sigma(i))$ ;  $C_j \leftarrow (j, L(j), \Sigma(j))$   
 Return  $(C_i, C_j, \delta)$  as the signature of  $\{i, j\}$ .

We refer to  $(l, L(l), \Sigma(l))$  as a *certificate* of node  $l$ .

- The verification algorithm **TVf**, on input  $tpk = (N, spk)$ , nodes  $i, j$  and a candidate signature  $\sigma$ , proceeds as follows:  
 If  $j < i$  then  $l \leftarrow j$ ;  $j \leftarrow i$ ;  $i \leftarrow l$  // swap  $i$  and  $j$  if necessary  
 Parse  $\sigma$  as  $(C_i, C_j, \delta)$ , parse  $C_i$  as  $(i, L_i, \Sigma_i)$ , parse  $C_j$  as  $(j, L_j, \Sigma_j)$   
 If  $\text{SVf}(spk, i\|L_i, \Sigma_i) = 0$  or  $\text{SVf}(spk, j\|L_j, \Sigma_j) = 0$  then return 0  
 If  $L_i \equiv \delta^2 L_j \bmod N$  then return 1 else return 0.
- The composition algorithm **Comp** takes nodes  $i, j, k$ , a signature  $\sigma_1 = (C_1, C_2, \delta_1)$  of  $\{i, j\}$  and a signature  $\sigma_2 = (C_3, C_4, \delta_2)$  of  $\{j, k\}$ , and proceeds as follows:

Let  $C_i \in \{C_1, C_2\}$  be such that  $C_i$  parses as  $(i, L_i, \Sigma_i)$ <sup>2</sup>  
 Let  $C_j \in \{C_1, C_2\}$  be such that  $C_j$  parses as  $(j, L_j, \Sigma_j)$   
 If  $C_j \notin \{C_3, C_4\}$  then return  $\perp$   
 Let  $C_k \in \{C_3, C_4\}$  be such that  $C_k$  parses as  $(k, L_k, \Sigma_k)$   
 If  $j < i < k$  then  $\delta \leftarrow \delta_1^{-1} \delta_2 \bmod N$ ; Return  $(C_i, C_k, \delta)$   
 If  $i < j < k$  then  $\delta \leftarrow \delta_1 \delta_2 \bmod N$ ; Return  $(C_i, C_k, \delta)$   
 If  $i < k < j$  then  $\delta \leftarrow \delta_1 \delta_2^{-1} \bmod N$ ; Return  $(C_i, C_k, \delta)$   
 If  $j < k < i$  then  $\delta \leftarrow \delta_1 \delta_2^{-1} \bmod N$ ; Return  $(C_k, C_i, \delta)$   
 If  $k < j < i$  then  $\delta \leftarrow \delta_1 \delta_2 \bmod N$ ; Return  $(C_k, C_i, \delta)$   
 If  $k < i < j$  then  $\delta \leftarrow \delta_1^{-1} \delta_2 \bmod N$ ; Return  $(C_k, C_i, \delta)$

A proof by induction can be used to show the following.

**Proposition 1.** *The FBTS-1 transitive signature scheme described above satisfies the correctness requirement of Definition 1.*

The proof is provided in [3]. We note that it was to ensure that this correctness requirement is met that we have been detailed regarding the specification of the composition algorithm above. We point in particular to the fact that the composition algorithm checks that the certificate  $C_j$  in the given signature of  $\{i, j\}$  exactly matches the one in the given signature of  $\{j, k\}$ . This ensures that

<sup>2</sup> This means that we assign the name  $C_i$  to whichever of  $C_1, C_2$  has its first component equal to the integer  $i$ . It is understood that if this is not possible then the algorithm halts and returns  $\perp$ . The same is true for the other similar steps that follow.

the public labels in these two certificates match, which is important in the proof of Proposition [□](#)

**ELIMINATING STATE.** The signing algorithm of the FBTS-1 scheme is stateful. It is important for composition that the signer associates a single public label to node  $i$ , and it is important for security that it associates to this a single secret label  $\ell(i)$ . (Else it would soon give away two different square roots of  $L(i)$ .) The MRTS, FBTS-2 and RSATS-1 schemes also have stateful signing algorithms, pointing to the fact that the natural formulation of many transitive signature schemes is in terms of a stateful signing algorithm. However, we note here that a simple transformation can be used to make the signer stateless, if so desired, without loss of security. Namely, let the signer's secret key include a key  $K$  specifying an instance  $F_K$  from a pseudorandom function family  $F$  [\[8\]](#), and use  $F_K(i)$  as the underlying coins (randomness) for all choices made by the signer related to node  $i$ . This enables the signer to recompute quantities as it needs rather than store them and yet be consistent, always creating the same quantities for a given node. Having pointed this out, however, in the rest of the paper we continue to work with stateful signing algorithms, since they are more natural and convenient in this context.

**COMPUTATIONAL COSTS.** The cost for the signature algorithm is dominated by multiplications and inversions modulo  $N$ , for both of which there exist algorithms quadratic in  $|N|$ , and the cost of generating two standard signatures, which depends on the choice of underlying standard signature scheme. It is not strictly necessary to test membership in  $\mathbb{Z}_N^*$ , because it is very unlikely that a randomly generated value is not coprime with  $N$ . Verification takes a couple of multiplications mod  $N$  and two standard signature verifications. The composition of two signatures involves one multiplication and possibly an inversion in  $\mathbb{Z}_N^*$ .

**SECURITY.** Forging a signature for FBTS-1 is trivial if an insecure instance is used for the modulus generator MG or the standard signature scheme SDS. The following theorem, however, states that the construction of FBTS-1 contains no weaknesses other than those induced by the underlying primitives.

**Theorem 1.** *Let MG be a modulus generator and let  $\text{SDS} = (\text{SKG}, \text{SSign}, \text{SVf})$  be a standard digital signature scheme. Let FBTS-1 be the transitive signature scheme associated to them as defined above. If the factoring problem associated to MG is hard and SDS is secure against forgery under adaptive chosen-message attack, then FBTS-1 is transitively unforgeable under adaptive chosen-message attack.*

We briefly sketch how a forgery for FBTS-1 can be used to either factor numbers generated by MG or break the underlying standard signature scheme SDS, and highlight a small but crucial detail for the analysis in a lemma. We refer to [\[3\]](#) for the full proof, that involves relatively standard reduction techniques and probability theory.

Signatures for FBTS-1 can be forged in only two ways: either there is the forgery that recycles node certificates from previously issued signatures, or there

is the forgery that includes at least one new node certificate. The latter can be easily transformed into an attack on SDS: the new node certificate is a successful forgery for SDS, because it contains a standard signature on a message that was not signed before. A forgery of the first type provides the signer with an edge label  $\delta'$  that is valid relative to the same public labels  $L(i')$  and  $L(j')$  he once issued for nodes  $i'$  and  $j'$  himself. (During this analysis, we assume wlog that  $i' < j'$ . If this is not the case, one can swap  $i'$  and  $j'$ .) Because these were computed as the squares of private labels  $\ell(i')$  and  $\ell(j')$ , he now knows two square roots of  $L(i') \cdot L(j')^{-1} \bmod N$ , namely  $\delta'$  and  $\delta \equiv \ell(i') \cdot \ell(j')^{-1} \bmod N$ .

It is tempting to say that since  $\ell(i')$  and  $\ell(j')$  were chosen at random, with probability one in two the signer now has two square roots  $\delta$  and  $\delta'$  such that  $\delta \not\equiv \pm\delta' \bmod N$ , enabling him to factor  $N$ . This argument would be correct if the forger only knew  $L(i')$  and  $L(j')$ , without having any further information on exactly which root the signer knows. However, by signing edges involving nodes  $i'$  or  $j'$ , the signer might have given away some additional information about his choices for  $\ell(i')$  and  $\ell(j')$ . It is crucial to the security of the scheme that this information doesn't help the forger in creating a forgery with edge label  $\delta' \equiv \pm\delta$ , as this would annihilate the signer's advantage in factoring  $N$ . Fortunately, it turns out that the exact value of  $\delta$  remains information-theoretically hidden from the forger as long as  $\{i', j'\}$  is not in the transitive closure of the signed edges.

We will prove this fact using the information-theoretical argument that for every possible square-root  $\delta$  of  $L(i')L(j')^{-1} \bmod N$  there are exactly as many choices for the signer's private information  $\ell$  that generate the given forger view and have  $\ell(i')\ell(j')^{-1} \equiv \delta \bmod N$ . As the secret labels are chosen uniformly at random from  $\mathbb{Z}_N^*$ , this implies that the issued signatures don't leak any useful information about which root the signer has in mind.

We represent the signer's secret information by a random variable  $\ell$  distributed uniformly over  $\mathbf{Secrets} = \{\ell \mid \ell : V \rightarrow \mathbb{Z}_N^*\}$ . The forger's view consists of a function  $L$  assigning a square mod  $N$  to each node in  $V$ , and a function  $\Delta$  assigning an edge label in  $\mathbb{Z}_N^*$  to each edge in  $\tilde{E}$ . (We discard the standard digital signatures on the node certificates, as they are irrelevant for this analysis.) However, not just any pair of functions  $\langle L, \Delta \rangle$  can occur as the forger's view. We say that forger view  $\langle L, \Delta \rangle$  is *consistent* with  $\ell \in \mathbf{Secrets}$  (and vice versa that  $\ell$  is consistent with  $\langle L, \Delta \rangle$ ) if and only if

$$L(i) \equiv \ell(i)^2 \bmod N \quad \text{for all } i \in V \quad (1)$$

$$\Delta(i, j) \equiv \ell(i)\ell(j)^{-1} \bmod N \quad \text{for all } \{i, j\} \in \tilde{E}, i < j \quad (2)$$

The set of all possible forger views  $\mathbf{Views}$  can then be defined as the set of all pairs  $\langle L, \Delta \rangle$  that are consistent with some  $\ell \in \mathbf{Secrets}$ . The actual view of the forger is a random variable **View** distributed over  $\mathbf{Views}$  as induced by  $\ell$ . The following lemma states that for every  $\langle L, \Delta \rangle \in \mathbf{Views}$  and for every  $\{i', j'\} \notin \tilde{E}$ , any square root  $\delta$  of  $L(i')L(j')^{-1} \bmod N$  is equally likely to be  $\delta \equiv \ell(i')\ell(j')^{-1} \bmod N$  when given only **View** =  $\langle L, \Delta \rangle$ , and hence that no forger, on input only **View**, can predict  $\delta$  with higher probability of success than random guessing. The following lemma formalizes this idea and is proven in the full version of this paper [3].

**Lemma 1.** *For any  $\langle L, \Delta \rangle \in \mathbf{Views}$ , for any  $\{i', j'\} \notin \widetilde{E}$  and for any  $\delta \in \mathbb{Z}_N^*$  with  $\delta^2 \equiv L(i')L(j')^{-1} \pmod{N}$ :*

$$\Pr[\delta \equiv \delta \pmod{N} \mid \mathbf{View} = \langle L, \Delta \rangle] = \frac{1}{4}. \blacksquare$$

## 4 Eliminating Node Certificates via Hashing

**THE IDEA.** The MRTS and FBTS-1 schemes rely on an underlying standard digital signature scheme to convince the verifier that the public label  $L(i)$  was associated to node  $i$  by the signer, and was not generated by some fraudulent third party. The disadvantage of this approach is that the signer has to provide the verifier with all necessary node certificates, thereby increasing the signature size as well as the computational cost for signing and verifying. In this section we show how the need for node certificates can be eliminated by specifying the public labels  $L(i)$  via the output of a hash function on input  $i$ . No explicit certification is attached to this value. Rather, we will be able to show that the edge label provides an “implicit authentication” of the associated node label that suffices to be able to prove that the scheme is transitively unforgeable under adaptive chosen-message attack assuming the hardness of factoring, in a model where the hash function is based on a random oracle.

**THE HASH FUNCTION.** The first thought regarding transforming FBTS-1 based on this idea is to simply let  $L(i) = H(i)$  where  $H$  is some public hash function. However,  $L(i)$  needs to be a quadratic residue in  $\mathbb{Z}_N^*$ , where  $N$  is the signer’s modulus, and this needs to be verifiable given  $N$  alone. In practice  $H$  must be built via a cryptographic hash function like SHA-1, which returns 160 bits. Standard techniques [5] can be used to build  $H$  from  $h$  so that it has range  $\mathbb{Z}_N^*$ , exploiting the fact that  $\mathbb{Z}_N^*$  is dense in  $\{0, 1\}^k$  where  $2^{k-1} \leq N < 2^k$  and that membership in  $\mathbb{Z}_N^*$  can be tested in  $\text{poly}(k)$  time given  $N$ . However, given that no polynomial-time algorithm to test quadratic residuosity is known, there is no practical way to ensure that  $H(i)$  is a quadratic residue while being able to verify this given  $N$ .

We could set  $L(i) = H(i)^2 \pmod{N}$  but this reveals a square-root of  $L(i)$  which makes the scheme insecure. Instead, reusing ideas from [11] and [13], we let the signer choose  $N$  to be a Blum integer (i.e.  $N = pq$  with  $p$  and  $q$  primes such that  $p \equiv q \equiv 3 \pmod{4}$ ). Then it is well-known that exactly one square-root (called the principal one) of each square is itself a square mod  $N$ . As a consequence, every square mod  $N$  is also a fourth power mod  $N$ , and has exactly four fourth roots. Now we will choose  $L(i), \ell(i)$  such that  $L(i) \equiv H(i)^2 \equiv \ell(i)^4 \pmod{N}$  where  $H$  is a hash function with range  $\mathbb{Z}_N^*[+1]$ , the elements of  $\mathbb{Z}_N^*$  with Jacobi symbol  $+1$ . Since the Jacobi symbol can be computed in polynomial time given  $N$ , such a hash function can be easily built starting from a cryptographic hash function.

**THE FBTS-2 SCHEME.** A modulus generator BG (as defined in Section 3), is said to be a *Blum modulus generator* if the primes  $p, q$  satisfy  $p \equiv q \equiv 3 \pmod{4}$ .

We associate to any given Blum modulus generator BG a transitive signature scheme FBTS-2 = (TKG, TSign, TVf, Comp) defined as follows:

- TKG, on input  $1^k$ , runs  $\text{BG}(1^k)$  to obtain  $(N, p, q)$  and outputs  $tpk = N$  as the public key and  $tsk = (N, p, q)$  as the matching secret key. All the following algorithms are now assumed to have oracle access to a function  $H_N: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*[+1]$ .
- TSign maintains state  $(V, \ell)$  where  $V \subseteq \mathbb{N}$  is the set of all queried nodes and the function  $\ell: V \rightarrow \mathbb{Z}_N^*$  assigns to each node  $i \in V$  a secret label  $\ell(i) \in \mathbb{Z}_N^*$ . When invoked on inputs  $tsk, i, j$ , meaning when asked to produce a signature on edge  $\{i, j\}$ , it does the following:

If  $j < i$  then  $l \leftarrow j$ ;  $j \leftarrow i$ ;  $i \leftarrow l$  // swap  $i$  and  $j$  if necessary

If  $i \notin V$  then

$V \leftarrow V \cup \{i\}$ ;  $L(i) \leftarrow H_N(i)^2 \bmod N$ ;  $\ell(i) \xleftarrow{R} L(i)^{\frac{1}{4}} \bmod N$

If  $j \notin V$  then

$V \leftarrow V \cup \{j\}$ ;  $L(j) \leftarrow H_N(j)^2 \bmod N$ ;  $\ell(j) \xleftarrow{R} L(j)^{\frac{1}{4}} \bmod N$

$\delta \leftarrow \ell(i)\ell(j)^{-1} \bmod N$

where the notation  $x \xleftarrow{R} (y)^{\frac{1}{4}} \bmod N$  means that  $x$  is chosen at random from all fourth roots of  $y \bmod N$ . (These roots can be efficiently computed using the prime factors  $p$  and  $q$ .) Return  $\delta$  as the signature on  $\{i, j\}$ .

- TVf, on input  $tpk = N$ , nodes  $i, j$  and a signature  $\delta$ , first swaps  $i$  and  $j$  if  $j < i$ . It returns 1 if  $H_N(i)^2 \equiv \delta^4 H_N(j)^2 \bmod N$  and returns 0 otherwise.
- Comp on input nodes  $i, j, k$  and signatures  $\delta_1, \delta_2$ , proceeds as follows:

If  $j < i$  then  $\delta_1 \leftarrow \delta_1^{-1} \bmod N$  ; If  $k < j$  then  $\delta_2 \leftarrow \delta_2^{-1} \bmod N$

$\delta \leftarrow \delta_1 \cdot \delta_2 \bmod N$

and outputs  $\delta$  as the transitive composition.

A proof by induction can be used to show the following.

**Proposition 2.** *The FBTS-2 transitive signature scheme described above satisfies the correctness requirement of Definition 1.*

**COMPUTATIONAL COSTS.** Since half of the elements in  $\mathbb{Z}_N^*$  have Jacobi symbol  $+1$ , a hash function evaluation requires the computation of two Jacobi symbols on average, which takes time quadratic in  $|N|$ . Computing square roots, however, is cubic in  $|N|$ , so the computation of the fourth roots (by extracting square roots twice) will dominate the cost of generating signatures. Verification and composition of signatures involve multiplications, inverses and Jacobi symbols mod  $N$ , all of which are operations quadratic in  $|N|$ .

**SECURITY.** In [3], we prove breaking the FBTS-2 scheme equivalent to factoring in the random oracle model. This means that in the experiment  $\text{Exp}_{\text{FBTS-2}, F}^{\text{tu-cma}}(k)$  used to define the advantage of an adversary  $F$ , the function  $H_N$  is assumed to be chosen at random from the space of all functions mapping  $\{0, 1\}^*$  to  $\mathbb{Z}_N^*[+1]$ . The result is stated as a theorem below.

**Theorem 2.** *Let BG be a Blum modulus generator. Let FBTS-2 be the transitive signature scheme as defined above. If the factoring problem associated to BG is hard, then FBTS-2 is transitively secure against forgery under adaptive chosen-message attack in the random oracle model.*

## 5 Transitive Signatures Based on RSA

In [12], Micali and Rivest mentioned the following scheme as a simpler scheme that can only be proven secure against a static forger, meaning that the forger must commit to all of his oracle queries before seeing the responses to any of them. While we still don't know how to prove security against an adaptive forger assuming only the one-wayness of RSA (and whether this can be done at all), we revisit the scheme here to prove it secure under the assumption that the one-more RSA-inversion problem, as described in the introduction, is hard.

In analogy with the modulus generator of the previous section, we define an *RSA key generator* RG as a randomized, polynomial-time algorithm that on input  $1^k$  outputs a tuple  $(N, e, d)$  where  $2^{k-1} \leq N < 2^k$  and  $ed \equiv 1 \pmod{\varphi(N)}$ . We do not restrict the type of generator, but only assume that its associated one-more RSA-inversion problem is hard.

**THE RSATS-1 SCHEME.** We associate to any RSA key generator RG and to any standard digital signature scheme  $\text{SDS} = (\text{SKG}, \text{SSign}, \text{SVf})$  a transitive signature scheme  $\text{RSATS-1} = (\text{TKG}, \text{TSign}, \text{TVf}, \text{Comp})$  defined as follows:

- TKG runs  $\text{SKG}(1^k)$  to generate a key pair  $(\text{spk}, \text{ssk})$  for SDS and runs  $\text{RG}(1^k)$  to generate an RSA key  $(N, e, d)$ . It outputs  $\text{tpk} = (N, e, \text{spk})$  as the public key and  $\text{tsk} = (N, d, \text{ssk})$  as the matching secret key.
- The signing algorithm  $\text{TSign}$  is identical to that of the FBTS-1 scheme, except that now the public label  $L(i)$  for node  $i$  is computed as  $L(i) \equiv \ell(i)^e \pmod{N}$ . The state information kept, the way of creating node certificates and the way of constructing the signature remain unchanged.
- The verification algorithm  $\text{TVf}$  is also very similar to that of FBTS-1: the only difference is the test on the edge label, which now consists of checking that  $L_i \equiv \delta^e L_j \pmod{N}$ .
- The  $\text{Comp}$  algorithm is perfectly identical to the composition algorithm of FBTS-1.

The scheme described above can be shown to satisfy the correctness requirement of Definition 1 using a proof by induction.

**COMPUTATIONAL COSTS.** Depending on the actual implementation of RSA, its computational overhead probably dominates over quadratic-time operations such as multiplications and inverses mod  $N$ . The generation of a transitive signature needs in the worst case two RSA encryptions, and two standard signatures for the node certificates. Signature verification takes one RSA encryption and two standard signature verifications, while quadratic operations are predominant in the composition algorithm.



**SECURITY OF RSATS-1.** The security analysis for this scheme against an adaptive forger is very similar to the analysis of FBTS-1, except that this time the certificate-recycling type of forgery can be proven equivalent to solving the one-more RSA-inversion problem associated to RG. The proof for the following theorem is given in [3].

**Theorem 3.** *Let RG be an RSA key generator and let  $\text{SDS} = (\text{SKG}, \text{SSign}, \text{SVf})$  be a standard digital signature scheme. Let RSATS-1 be the transitive signature scheme as defined above. If the one-more RSA-inversion problem associated to RG is hard and SDS is secure against forgery under adaptive chosen-message attack, then RSATS-1 is transitively secure against forgery under adaptive chosen-message attack.*

**THE RSATS-2 SCHEME.** The idea of replacing node certificates by a suitable hash function can also be applied to the RSATS-1 scheme. Since this time the public labels are uniformly distributed over the whole of  $\mathbb{Z}_N^*$ , we can use a hash function  $H_N : \mathbb{N} \rightarrow \mathbb{Z}_N^*$  to directly map node  $i$  to its public label  $L(i) = H_N(i)$ . The unambiguous invertibility of RSA encryption allows for the first completely stateless signature algorithm: the signature for edge  $\{i, j\}$  (swapping  $i$  and  $j$  if  $j < i$ ) is computed as  $\delta \equiv (H_N(i) \cdot H_N(j)^{-1})^d \bmod N$ . The verification of signature  $\delta$  for edge  $\{i, j\}$ ,  $i < j$ , is done by checking that  $H_N(i) \equiv \delta^e H_N(j) \bmod N$ . Composition of signatures works as in the FBTS-2 scheme by multiplying the (if necessary inverted) edge labels. The proofs of correctness and security (in the random oracle model, assuming that the one-more RSA-inversion problem associated to RG is hard) are very similar to those given for RSATS-1 and were hence omitted.

## Acknowledgments

Mihir Bellare is supported in part by NSF grant CCR-0098123, NSF grant ANR-0129617, and an IBM Faculty Partnership Development Award. Gregory Neven is supported by a Research Assistantship and a travel credit from the Fund for Scientific Research, Flanders (Belgium) (F.W.O.–Vlaanderen). Work done while Gregory was visiting UCSD.

## References

1. M. ABDALLA AND L. REYZIN. A new forward-secure digital signature scheme. *Advances in Cryptology – ASIACRYPT '00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.
2. M. BELLARE, C. NAMPREMPRE, D. POINTCHEVAL AND M. SEMANKO. The One-More-RSA-Inversion problems and the security of Chaum's blind signature scheme. Cryptology ePrint Archive: Report 2001/002, <http://eprint.iacr.org/2001/002/>. Preliminary version, entitled "The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme," in *Financial Cryptography '01*, Lecture Notes in Computer Science Vol. 2339, P. Syverson ed., Springer-Verlag, 2001.

3. M. BELLARE AND G. NEVEN. Transitive signatures based on factoring and RSA. Full version of this abstract, available via <http://www-cse.ucsd.edu/users/mihir>.
4. M. BELLARE AND A. PALACIO. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Computer Science Vol. 2442, M. Yung ed., Springer-Verlag, 2002.
5. M. BELLARE AND P. ROGAWAY. Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of the 1st Annual Conference on Computer and Communications Security*, ACM, 1993.
6. S. CHARI, T. RABIN AND R. RIVEST. An efficient signature scheme for route aggregation. Manuscript, February 2002.  
<http://theory.lcs.mit.edu/~rivest/publications.html>.
7. D. CHAUM. Blind signatures for untraceable payments. *Advances in Cryptology – CRYPTO 82 Proceedings*, D. Chaum, R. Rivest and A. Sherman eds., Plenum Press.
8. O. GOLDBREICH, S. GOLDWASSER AND S. MICALI. How to construct random functions. *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.
9. S. GOLDWASSER, S. MICALI AND R. RIVEST. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, Vol. 17, No. 2, April 1988, pp. 281–308.
10. L. GUILLOU AND J. J. QUISQUATER. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. *Advances in Cryptology – CRYPTO '88*, Lecture Notes in Computer Science Vol. 403, S. Goldwasser ed., Springer-Verlag, 1988.
11. R. JOHNSON, D. MOLNAR, D. SONG AND D. WAGNER. Homomorphic signature schemes. *Topics in Cryptology – CT-RSA '02*, Lecture Notes in Computer Science Vol. 2271, B. Preneel ed., Springer-Verlag, 2002.
12. S. MICALI AND R. RIVEST. Transitive signature schemes. *Topics in Cryptology – CT-RSA '02*, Lecture Notes in Computer Science Vol. 2271, B. Preneel ed., Springer-Verlag, 2002.
13. S. MICALI AND L. REYZIN. Improving the exact security of digital signature schemes. *Journal of Cryptology*, Vol. 15, Number 1, 2002, pp. 1–18.
14. R. RIVEST. Two signature schemes. Slides from talk given at Cambridge University, October 17, 2000.  
<http://theory.lcs.mit.edu/~rivest/publications.html>.
15. R. STEINFELD, L. BULL AND Y. ZHENG. Content Extraction Signatures. *Information Security and Cryptology – ICISC 2001*, Lecture Notes in Computer Science Vol. 2288, K. Kim ed., Springer-Verlag, 2002.

# 1-out-of- $n$ Signatures from a Variety of Keys

Masayuki Abe<sup>1</sup>, Miyako Ohkubo<sup>2</sup>, and Koutarou Suzuki<sup>1</sup>

<sup>1</sup> NTT Laboratories,

1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan,  
{abe,koutarou}@isl.ntt.co.jp

<sup>2</sup> Chuo University,

1-13-27, Kasuga, Bunkyo-ku, Tokyo 112-8551 Japan,  
omiyako@apricot.ocn.ne.jp

**Abstract.** This paper addresses how to use public-keys of several different signature schemes to generate 1-out-of- $n$  signatures. Previously known constructions are for either RSA-keys only or DL-type keys only. We present a widely applicable method to construct a 1-out-of- $n$  signature scheme that allows mixture use of different flavors of keys at the same time. The resulting scheme is more efficient than previous schemes even if it is used only with a single type of keys. With all DL-type keys, it yields shorter signatures than the ones of the previously known scheme based on the witness indistinguishable proofs by Cramer, et al. With all RSA-type keys, it reduces both computational and storage costs compared to that of the Ring signatures by Rivest, et al.

## 1 Introduction

A 1-out-of- $n$  signature convinces a verifier that a document is signed by one of  $n$  possible independent signers without allowing the verifier to identify which signer it was. It can be seen as a simple group signature that has no group manager who can revoke the identity of the signer in case of emergency. Such a signature can also be seen as a kind of non-interactive proof that the signer owns a witness (secret-key) that corresponds to one of  $n$  commitments (public-keys) or theorems without leaking which one it really is. Such a primitive, as a signature scheme and/or a proof system, plays a central role in variety of applications such as group signatures [8,5], designated verifier signatures [17], mix-nets [1], electronic voting [10,11] and so on.

In [9], Cramer, Damgård and Shoenmakers presented a widely applicable yet efficient construction of  $t$ -out-of- $n$  witness indistinguishable proofs [13] based on secret sharing and public-coin honest verifier zero-knowledge proofs. It can be transformed into  $t$ -out-of- $n$  signatures via the Fiat-Shamir technique [12]. It is especially suitable for converting Schnorr signatures [23] and Guillou-Quisquater signatures [16] into  $t$ -out-of- $n$  signatures. It also allows to involve RSA signature scheme based on a zero-knowledge proof of knowledge about the factors of RSA modulus, e.g. [7,6], but they are less efficient than the Schnorr or the GQ signatures both in computation and storage. [22] offers more intricate construction of  $t$ -out-of- $n$  proofs for membership.

In [21], an efficient construction of 1-out-of- $n$  signatures with RSA public-keys was introduced by Rivest, Shamir and Tauman. Called the Ring Signature Scheme, it is based on trapdoor one-way permutations (TPs for short) and an ideal block cipher that is regarded as a perfectly random permutation. The name reflects its unique structure such that a signer who knows at least one witness (trapdoor information) can connect the head and tail of a series of  $n$  random permutations to shape the sequence into a ring. Since the trapdoor is essential in their construction, it is only for the keys like RSA's and the discrete-log keys are not supported.

There are other solutions that are more efficient but work only in non-separable models where all public-keys are related. For instance, when public-keys of the Schnorr signature scheme are chosen from a common group, one can construct an efficient 1-out-of- $n$  signature scheme as shown in Appendix A. Such non-separable but highly efficient schemes may be useful when used within a specific members. In general, however, public-keys are selected independently by each signer. Even key-length would differ from user to user. Constructions based on [9] and [21] suit a separable model where no underlying group are assumed. Hence, they are 'setup-free'; if one utilizes an existing public-key infrastructure, the key-setup phase only for this purpose is unnecessary. Furthermore, each key can be freely updated whenever each user wishes.

As introduced in [21], one application of 1-out-of- $n$  signatures is to involve somebody else's public-keys into one's signature without their agreement. Although there are pros and cons for such usage, it is surely useful for protecting privacy. Unfortunately, all above mentioned known schemes have particular shortcomings for this purpose; What if one is using a DL type public-key while others are using RSA? Generating a new RSA key only for this purpose is not a great idea. It is important to have wide freedom for choosing various public-keys to involve.

**Our Contribution.** We present a widely applicable method of constructing 1-out-of- $n$  signature schemes that allows to use several flavours of public-keys such as these for integer factoring based schemes and discrete-log based schemes at the same time. We describe two classes of signature schemes, which we call trapdoor-one-way type and three-move type, whose public-keys can be used for our construction.

Our approach also has several advantages even for the use with the same kind of keys like the former schemes:

- When our scheme is used only with public-keys of three-move type signature schemes converted from zero-knowledge proof system, it results in a more efficient scheme than previously known three-move based construction [9] with regard to the size of signatures. For large  $n$ , it saves signature length about *by half*. Since this type of schemes includes the discrete-log based public-keys, this can be seen as the first construction of a ring signature scheme based on the discrete logarithm problem.
- When our scheme is used only with the trapdoor-one way based public-keys such as RSA, it results in a simplified ring signature scheme. By eliminat-

ing the use of block cipher and costly domain adjustment from the former scheme [21], our scheme offers shorter signature and less computation. In particular,

- The signature size of ours is about 20% less than that of the previous construction when RSA with modulus size 1024bits are used.
- Both size of signature and computation in our signature generation is proportional to the *average* size of the modulus while that of former scheme it is proportional to the *maximum* size of the modulus. Accordingly, one long modulus does not impact efficiency in our scheme unlike the previous scheme.

We will show several concrete examples following an abstract construction. The security is proven in the random oracle model [3] as well as previously known schemes.

The rest of this paper is organized as follows. Section 2 defines security of 1-out-of-n signatures. We review two constructions that work in the separable model in Section 3. Section 4 describes our construction in an abstract way. Some concrete examples are given in Section 5. It includes a discrete-log version of the ring signature scheme, improved and simplified version of the RSA-based ring signatures, and small case of mixture use of RSA and DL type signatures. In Section 6 the efficiency of some concrete instantiations are analyzed in detail.

## 2 Security Definitions

We first of all define 1-out-of- $n$  signature scheme as follows.

**Definition 1. (Syntax).** A 1-out-of- $n$  signature scheme,  $\mathbf{S}^{1,n}$ , is a triple of polynomial-time algorithms,  $\mathbf{S}^{1,n} = (\mathcal{G}^{1,n}, \mathcal{S}^{1,n}, \mathcal{V}^{1,n})$ :

- $(sk, vk) \leftarrow \mathcal{G}^{1,n}(1^\kappa)$  A probabilistic algorithm that takes security parameter  $\kappa$  and outputs private key  $sk$  and public-key  $vk$ .
- $\sigma \leftarrow \mathcal{S}_{sk}^{1,n}(m, L)$  A (probabilistic) algorithm that takes message  $m$ , and a list, say  $L$ , of public-keys including the one that corresponds to  $sk$ , outputs signature  $\sigma$ .
- $1/0 \leftarrow \mathcal{V}_L^{1,n}(m, \sigma)$  An algorithm that takes message  $m$  and signature  $\sigma$ , and outputs either 1 or 0 meaning *accept* and *reject*, respectively. We require that  $\mathcal{V}_L^{1,n}(m, \mathcal{S}_{sk}^{sig}(m, L)) = 1$  for any message  $m$ , any  $(sk, vk)$  generated by  $\mathcal{G}^{1,n}$ , and any  $L$  that includes  $vk$ .

Note that  $\mathcal{G}^{1,n}$  does not generate  $L$  but each key pairs. Therefore, if  $L$  includes public-keys based on different security parameters, the security of  $\mathbf{S}^{1,n}$  is set to the smallest one among them. As we will see,  $L$  can include several types of public-keys all at the same time such as for RSA and Schnorr in a particular construction.  $\mathcal{G}^{1,n}$  may be extended to take a description of the key-type to support such variety flavour of key pairs. By  $|L|$ , we denote the number of public-keys in  $L$  hereafter.

The security of 1-out-of- $n$  signature schemes has two aspects: *Signer ambiguity* and *Unforgeability*. Informally, the signer ambiguity is that it is infeasible to identify which signing key is used to generate a signature.

**Definition 2. (Signer Ambiguity).** Let  $L = \{vk_1, \dots, vk_n\}$  where each key is generated as  $(vk_i, sk_i) \leftarrow \mathcal{G}^{1,n}(1^{\kappa_i})$ .  $\mathbf{S}^{1,n}$  is perfectly signer-ambiguous if, for any  $L$ , any message  $m$ , and any  $\sigma$  generated by  $\sigma \leftarrow \mathcal{S}_{sk}^{1,n}(m, L)$  where  $sk \leftarrow \{sk_1, \dots, sk_n\}$ , given  $(L, m, \sigma)$ , any unbound adversary  $\mathcal{A}$  outputs  $i$  such that  $sk = sk_i$  with probability exactly  $1/|L|$ .

Here,  $a \leftarrow b$  denotes a uniform choice of an element from set  $b$  and its assignment to  $a$ . It is important to see that unbound adversary can compute all private keys from  $L$ . In practice, it means that when each public-key is owned by an independent party, they remain uncertain who else has issued a signature involving their public-keys.

Unforgeability of 1-out-of- $n$  signature scheme is defined by naturally extending the notion of existential unforgeability against adaptive chosen message attacks (EUF-CMA) [15], which is the strongest security for ordinary signature schemes. In chosen message attacks, an adversary is given unbound access to the signing oracle and allowed to ask signatures for arbitrary messages. To adapt to our situation, we further allow the adversary to choose arbitrary set of public-keys as a subset of initially considered set of public-keys every time it access to the signing oracle. It is stressed that one can generate 1-out-of- $n$  signatures for any message and any list of public-keys as long as it includes one's own key. So the definition of unforgeability should not treat “append-your-own-key-then-forge” activities as a forgery. Formal definition is as follows.

**Definition 3. (Existential Unforgeability against Adaptive Chosen Message and Chosen Public-Key Attacks).** Let  $(vk_i, sk_i) \leftarrow \mathcal{G}^{1,n}(1^{\kappa_i})$  for  $i = 1, \dots, n$ . Let  $\kappa = \min(\kappa_1, \dots, \kappa_n)$  and  $\mathcal{L} = \{vk_1, \dots, vk_n\}$ . Let  $\mathcal{SO}_{\mathcal{L}}^{1,n}(m_i, L_i)$  be a signing oracle that takes any message  $m \in \{0, 1\}^*$  and any  $L_i \subseteq \mathcal{L}$  and outputs a valid signature  $\sigma_i$  that satisfies  $\mathcal{V}_{L_i}^{1,n}(m_i, \sigma_i) = 1$ . We say  $\mathbf{S}^{1,n}$  is existentially unforgeable against adaptive chosen message and chosen public-key attacks if, for any polynomial-time oracle machine  $\mathcal{A}$  such that  $(L, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{SO}_{\mathcal{L}}^{1,n}(\cdot, \cdot)}(\mathcal{L})$ , its output satisfies  $\mathcal{V}_L^{1,n}(m, \sigma) = 1$  only with negligible probability in  $\kappa$ . Restriction is that  $L \subseteq \mathcal{L}$  and  $(L, m, \sigma) \notin \{(L_i, m_i, \sigma_i)\}$  where  $\{(L_i, m_i, \sigma_i)\}$  is the set of oracle queries and replies between  $\mathcal{A}$  and  $\mathcal{SO}_{\mathcal{L}}^{1,n}$ .

The above definition is a seamless extension of EUF-CMA since the case of  $n = 1$  is equivalent to that. Note that the size of  $\mathcal{L}$  can be a security parameter as well, though it is not our case. It is important to see that the above definition states that the list of public-keys must not be altered as well as the message. That is, one should not be able to add or remove public-keys associated to given signatures.

### 3 Previous Schemes

#### 3.1 Witness Indistinguishable Signatures [9]

Here we review the witness indistinguishable signatures from [9] with a concrete discrete logarithm setting. Let  $p_i, q_i$  be large primes. Let  $\langle g_i \rangle$  denote a prime subgroup of  $\mathbb{Z}_{p_i}^*$  generated by  $g_i$  whose order is  $q_i$ . Let  $x_i, y_i$  be  $y_i = g_i^{x_i} \bmod p_i$ . Here,  $x_i$  is the secret-key and  $(y_i, p_i, q_i, g_i)$  is the public-key. Let  $L$  be a set of  $(y_i, p_i, q_i, g_i)$  for  $i = 0, \dots, n-1$ . Let  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be a publicly available hash function, where  $\ell$  is larger than the largest  $|q_i|$ .

A signer who owns secret key  $x_k$  generates a signature for message  $m$  with public-key list  $L$  that includes his own public-key, in the following way.

**W-1** (Simulation step): For  $i = 0, \dots, n-1, i \neq k$ , select  $s_i, c_i \leftarrow \mathbb{Z}_{q_i}$  and compute  $z_i = g_i^{s_i} y_i^{c_i} \bmod p_i$ .

**W-2** (Real proof step): Select  $r_k \leftarrow \mathbb{Z}_{q_k}$  and compute

$$\begin{aligned} z_k &= g_k^{r_k} \bmod p_k \\ c &= H(L, m, z_0, \dots, z_{n-1}) \\ c_k &= c \oplus (c_0 \oplus \dots \oplus c_{k-1} \oplus c_{k+1} \oplus \dots \oplus c_{n-1}) \quad (\oplus: \text{bitwise-XOR.}) \\ s_k &= r_k - c_k \cdot x_k \bmod q_k. \end{aligned}$$

The resulting signature is  $\sigma = (c_0, s_0, \dots, c_{n-1}, s_{n-1})$ . A  $(L, m, \sigma)$  is valid if

$$c_0 \oplus \dots \oplus c_{n-1} = H(L, m, g_0^{s_0} y_0^{c_0} \bmod p_0, \dots, g_{n-1}^{s_{n-1}} y_{n-1}^{c_{n-1}} \bmod p_{n-1}).$$

The size of  $\sigma$  is  $n\ell + \sum_{i=0}^{n-1} |q_i|$  bits.  $L$  does not necessarily contain whole public-keys but some identifiers of the keys. The security can be proven in the random oracle model by using the rewinding simulation technique [14,20,19].

#### 3.2 Ring Signatures with Trapdoor One-Way Permutations [21]

Let  $f_i : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$  be a trapdoor one-way permutation where its inverse,  $f_i^{-1}$ , can be computed only if the trapdoor information is known. Let  $E, D$  be a symmetric-key encryption and decryption function whose message space is  $\{0, 1\}^\ell$ . Let  $H$  be a hash function whose output domain matches to the key-space of  $E, D$ .

Given  $f_0, \dots, f_{n-1}$ , the signer who can compute  $f_k^{-1}$  generates a signature, for message  $m$  in the following way.

**R-1** (Initialization): Compute  $r_{n-1} = D_K(c_0)$  where  $K = H(m)$  and  $c_0 \leftarrow \{0, 1\}^\ell$ .

**R-2** (Forward sequence): For  $i = 0, \dots, k-1$ , compute  $c_{i+1} = E_K(c_i \oplus f_i(s_i))$  for  $s_i \leftarrow \{0, 1\}^\ell$ .

**R-3** (Backward sequence): For  $i = n-1, \dots, k+1$ , compute  $r_{i-1} = D_K(r_i \oplus f_i(s_i))$  for  $s_i \leftarrow \{0, 1\}^\ell$ .

**R-4** (Shaping into a ring): Compute  $s_k = f_k^{-1}(c_k \oplus r_k)$

The resulting signature is  $(c_0, s_0, s_1, \dots, s_{n-1})$ . A signature-message pair is verified by computing  $K = H(m)$  and  $c_{i+1} = E_K(c_i \oplus f_i(s_i))$  for  $i = 0, \dots, n-1$ , and accept if  $c_n = c_0$  holds.

In practice, each trapdoor permutations will be defined over individual domain such as  $\mathbb{Z}_{N_i}$ . In such a case, the above scheme need to transform such individual functions into common-domain trapdoor permutations. This transformation incurs some overhead. The following method is suggested in [21] to transform  $\mathcal{E}_i$  into  $f_i$  defined over common-domain  $\{0, 1\}^\ell$  where  $\ell = \max\{|N_i|\} + 160$ . Let  $\mathcal{E}_i$  be the RSA encryption function with modulus  $N_i$ . Let  $Q$  and  $S$  be positive integers such that  $QN_i + S = s$  and  $0 \leq S < N_i$ . Define

$$f_i(s) = \begin{cases} QN_i + \mathcal{E}_i(S) & \text{if } (Q+1)N_i \leq 2^\ell \\ s & \text{otherwise.} \end{cases}$$

In order for the latter case to happen only with negligible probability,  $\ell$  should be polynomially larger than the size of largest modulus. For instance, if the largest modulus is 2048 bits,  $\ell$  will be 2048 + 160 bits. Accordingly, the resulting signature size is  $2208(n+1)$  bits. This would be a large overhead when other moduli are all 1024bits.

The above ring signature is existentially unforgeable against adaptive chosen message attacks in the ideal cipher model where  $E$  and  $D$  are modeled by truly random permutations.

### 3.3 Other Related Works

[4] extends the scheme of [21] to a threshold scheme with the cost of  $O(2^t \log n)$  efficiency for threshold  $t$ . [18] considers deniable ring authentication that accepts variety of public-keys and a threshold of signers. It however, needs interaction between the signer and the verifier.

## 4 Our Scheme

### 4.1 Type of Keys and Signature Schemes

This section describes signature schemes whose public-key can be used to our construction of 1-out-of- $n$  signature scheme. Let  $\mathbf{S} = (\mathcal{G}^{\text{sig}}, \mathcal{S}^{\text{sig}}, \mathcal{V}^{\text{sig}})$  be a signature scheme. We require that underlying signature scheme be secure (existentially unforgeable) against adaptive chosen message attacks. For this to be achieved, it must be at least hard to compute  $sk$  from  $vk$ .

We consider two types of signature schemes which we call *Hash-then-One-Way type* (**type-H**) and *Three-move type* (**type-T**) in the rest of this paper.

A representative of **type-H** is the Full-domain RSA signature scheme. Let  $F$  be a trapdoor one-way function and  $I$  be its inverse function. For any  $c$  taken from appropriate domain, computing  $s = F_{vk}(c)$  is easy but any preimage of  $s$  cannot be computed in polynomial-time. Trapdoor information  $sk$  allows one to efficiently compute one of the pre-images of  $s$ . The signing function  $\mathcal{S}^{\text{sig}}$  involves



$I$  and a hash function  $H : \{0,1\}^* \rightarrow \Delta$  that hashes message  $m$  and auxiliary information if any. Domain  $\Delta$  is assumed to be an abelian group such as modulo an RSA composite that depends on the particular detail of the signature scheme and the security parameter.  $H$  can be a composition of hash functions. The verification function  $\mathcal{V}^{\text{sig}}$  of **type-H** consists of  $F$  and  $H$ .  $H$  is the same as that in  $\mathcal{S}^{\text{sig}}$ . By  $F$ , signature  $\sigma$  is transformed into an element of  $\Delta$  so that the result can be compared with the hashed message. In summary, **type-H** is as follows.

#### HASH-AND-ONE-WAY TYPE

##### SIGNING

$\mathcal{S}_{sk}^{\text{sig}}(m) =$   
 $c = H(m, aux)$   
 $s = I_{sk}(c)$   
 Return  $\sigma = (s, aux)$

##### VERIFICATION

$\mathcal{V}_{vk}^{\text{sig}}(m, \sigma) =$   
 $\sigma \xrightarrow{p} (s, aux) \text{ } (\xrightarrow{p}: \text{parsing})$   
 $c = H(m, aux)$   
 $e = F_{vk}(s)$   
 Return 1 if  $c = e$ . Otherwise, 0.

The security of **type-H** requires that computing  $I_{sk}(c)$  without  $sk$  be intractable. Precise description is as follows.

**Assumption 1 (Intractability of Computing  $I$ )** *For any probabilistic poly-time algorithm  $\mathcal{A}$ , for  $(vk, sk) \leftarrow \mathcal{G}^{\text{sig}}(1^\kappa)$ , and for  $c \leftarrow \Delta$ ,  $F_{vk}(\mathcal{A}(c, vk)) = c$  happens only with negligible probability in  $\kappa$ . Probability is taken over coin flips of  $\mathcal{A}$ ,  $\mathcal{G}^{\text{sig}}$ , and the choice of  $c$ .*

To prevent  $\mathcal{A}$  from being successful by random guess,  $I$  must not shrink  $\Delta$  into exponentially small domain. Typically,  $I$  is one-to-one with regard to variable  $c$ . Finally, note that the above intractability assumption for computing  $I$  is stronger than that of for computing  $sk$  only from  $vk$ .

Next we describe **type-T** schemes. As the name implies, this type of schemes are from three-move honest verifier zero-knowledge proofs. Classical Fiat-Shamir signature scheme belongs to this type. Here, signing function  $\mathcal{S}^{\text{sig}}$  involves three functions, say  $A$ ,  $H$ , and  $Z$  used in each stage of three-move honest verifier zero-knowledge proof system.  $A$  generates the first-move commitment  $a$  and with regard to randomness  $r$ .  $H$  is a hash function  $\{0,1\}^* \rightarrow \Delta$  used to generate a challenge string  $c$  from message  $m$  and commitment  $a$ .  $Z$  is an answer generation function that generates an answer, say  $s$ , to the challenge. The verification function  $\mathcal{V}^{\text{sig}}$  involves two functions  $V$  and  $H$ .  $V$  is a checking predicate of the embedded zero-knowledge proof system. It converts  $s$  and  $c$  into  $z$  which is supposed to equal to  $a$ . If it is the case, hashing  $z$  with message  $m$  by using  $H$  in the same way as in signing procedure outputs  $e$  that matches to  $c$ . Abstract description follows.

## THREE-MOVE TYPE

## SIGNING

$$\begin{aligned} \mathcal{S}_{sk}^{\text{sig}}(m) = \\ a &\leftarrow A(sk; r) \\ c &= H(m, a) \\ s &= Z(sk, r, c) \\ \text{Return } \sigma &= (s, c) \end{aligned}$$

## VERIFICATION

$$\begin{aligned} \mathcal{V}_{vk}^{\text{sig}}(m, \sigma) = \\ \sigma &\xrightarrow{p} (s, c) \\ z &= V(s, c, vk) \\ e &= H(m, z) \\ \text{Return } 1 &\text{ if } c = e. \text{ Otherwise } 0. \end{aligned}$$

The following property is assumed to **type-T** schemes.

**Definition 4. (Collision Property).** *There exists a polynomial-time algorithm that computes  $sk$  from  $(c, s, c', s')$  and  $vk$  where  $(c, s)$  and  $(c', s')$  are two unequal valid signatures that correspond to the same  $(a, m)$  given to hash function  $H$ .*

This property is frequently used for proving the security of **type-T** schemes such as Schnorr signatures and Modified ElGamal signatures and vast number of their variants.

Regardless of the types, we require that the signature scheme is simulatable in a particular model. Intuitively, it must be possible to construct a simulator that simulates the signing oracle without the signing key. In many schemes, this property is achieved in the random oracle model. Precise definition of this property is as follows.

**Definition 5. (Simulatability of S in the Random Oracle Model).** *A signature scheme,  $\mathbf{S}$ , is  $(t, \epsilon, q_s, q_h)$ -simulatable in the random oracle model if for any key-pair  $(sk, vk)$  generated by  $\mathcal{G}^{\text{sig}}(1^\kappa)$  and for any algorithm  $\mathcal{A}$  that refers random oracle  $H$  at most  $q_h$  times and  $\mathcal{S}_{sk}^{\text{sig}}$  at most  $q_s$  times, there exists a pair of interactive machines,  $\mathcal{M}^{\text{sim}} = (\mathcal{S}^{\text{sim}}, H^{\text{sim}})$ , that interacts with  $\mathcal{A}$  in such a way that the total running time is at most  $t$ , and statistical distance of the probability distribution of  $\text{view}_{\mathcal{A}}(vk, \mathcal{S}_{sk}^{\text{sig}}, H)$  and  $\text{view}_{\mathcal{A}}(vk, \mathcal{M}_{vk}^{\text{sim}})$  is at most  $2\epsilon$ . Here, the probability is taken over all coin flips of  $\mathcal{G}^{\text{sig}}$ ,  $\mathcal{S}^{\text{sig}}$ ,  $H$ ,  $\mathcal{M}^{\text{sim}}$ , and  $\mathcal{A}$ .*

The above definition can be generalized to deal with multiple oracles for signature schemes that involves multiple hash functions if necessary. Simulatability is featured in many practical EUF-CMA signature schemes such as the Schnorr signature scheme, and FDH-RSA scheme. Given this property, one can say that an event that happens with probability  $\mu$  in the real run also happens with probability at least  $\mu - \epsilon$  in the simulation.

## 4.2 Description

### [Key Generation]

A signer generates his own key pairs by using the signature generation function of a signature scheme of his choice:  $(sk, vk) \leftarrow \mathcal{G}^{\text{sig}}(1^\kappa)$

**[Public-Key Listing]**

Collect public-keys and list them in  $L$ . Then, insert  $vk$  to the randomly chosen position of the list. Let  $L = \{vk_0, \dots, vk_{n-1}\}$  where  $vk_k = vk$  for some  $k \in \{0, \dots, n-1\}$ . (Corresponding signing key  $sk$  is referred as  $sk_k$  hereafter.)

For  $a, b \in \Delta_i$ , let  $a + b$  denote the group operation of abelian group  $\Delta_i$  and  $a - b$  be the group operation with inverse of  $b$ . These binary operators are used without subscripts that denotes each group. Let  $H_i : \{0, 1\}^* \rightarrow \Delta_i$  be a hash function. Domain  $\Delta_i$  depends on  $vk_i$ .

**[Signature Generation]**

**G-1** (Initialization): Compute

$$e_k = \begin{cases} A_k(sk_k; \alpha) & (vk_k \text{ is type-T}), \text{ or} \\ \beta & (vk_k \text{ is type-H}), \end{cases}$$

where  $\alpha \leftarrow A_k$  and  $\beta \leftarrow \Delta_k$  ( $A_k$  denotes an appropriate space of randomness defined by the algorithm of  $A_k$  and  $sk_k$ ). Then compute  $c_{k+1} = H_{k+1}(L, m, e_k)$ .

**G-2** (Forward sequence): For  $i = k+1, \dots, n-1, 0, \dots, k-1$ , compute

$$e_i = \begin{cases} V_i(s_i, c_i, vk_i) & (vk_i \text{ is type-T}), \text{ or} \\ c_i + F_i(s_i, vk_i) & (vk_i \text{ is type-H}), \end{cases}$$

where  $s_i$  is randomly chosen. Then compute  $c_{i+1} = H_{i+1}(L, m, e_i)$ .

**G-3** (Forming the ring):

$$s_k = \begin{cases} Z_k(sk_k, \alpha, c_k) & (vk_k \text{ is type-T}), \text{ or} \\ I_k(\beta - c_k, sk_k) & (vk_k \text{ is type-H}). \end{cases}$$

The resulting signature for  $m$  and  $L$  is  $(c_0, s_0, s_1, \dots, s_{n-1})$ .

**[Signature Verification]**

For  $i=0, \dots, n-1$ , compute

$$e_i = \begin{cases} V_i(s_i, c_i, vk_i) & (vk_i \text{ is type-T}), \text{ or} \\ c_i + F_i(s_i, vk_i) & (vk_i \text{ is type-H}), \end{cases}$$

and then  $c_{i+1} = H_{i+1}(L, m, e_i)$  if  $i \neq n-1$ . Accept if  $c_0 = H_0(L, m, e_{n-1})$ . Reject otherwise.

**4.3 Remark on Compatibility of Keys**

Some signature schemes are neither **type-T** nor **type-H**. For such schemes, we consider *compatibility* among signature schemes. Signature scheme A is compatible with scheme B if 1) A's private and public keys can be used to issue and

verify signatures of scheme B, and 2) breaking B (in EUF-CMA sense) implies breaking A using the same key. For instance, DSS is not either type but it is compatible with the Schnorr signature scheme of **type-T**. Since breaking the Schnorr signature scheme implies that the discrete-log is tractable with regard to the key, it implies DSS is broken, too. Thus, DSS keys can be involved in our scheme with **type-T**.

With regard to **type-H** schemes, however, special care may be needed. Remember that **type-H** only shows the ability of computing  $I_{sk}(\cdot)$  and does not necessarily imply possession of  $sk$ . Therefore, it is not sufficient that scheme A's keys can be used to scheme B, but it has to be true that ability of computing  $I_{sk}(\cdot)$  of scheme B is sufficient to generate signatures of A.

The signature scheme in [2] is a curious scheme that belongs to **type-H** but its keys are also compatible with **type-T** ones. In such a case, one can select more efficient type to involve the keys.

## 5 Concrete Examples

Leaving out the security proof for the abstract scheme (which turns out to be similar to the one shown in Section 5.3), we present concrete examples and their security proofs in order to help readers who is familiar with RSA and Schnorr signatures grasp the ideas for our construction and the security proofs.

### 5.1 All Discrete-log Case

For  $i = 0, \dots, n-1$ , let  $(y_i, p_i, q_i, g_i)$  be DL public-keys as described in Section 3.1 and  $H_i : \{0, 1\}^* \rightarrow \mathbb{Z}_{q_i}$  be hash functions. Let  $L$  be a list of these public-keys. A signer who has private key  $x_k$  generates a signature for message  $m$  as follows.

#### [Signature Generation]

- D-1** (Initialization): Select  $\alpha \leftarrow \mathbb{Z}_{q_k}$  and compute  $c_{k+1} = H_{k+1}(L, m, g_k^\alpha \bmod p_k)$ .
- D-2** (Forward sequence): For  $i = k+1, \dots, n-1, 0, \dots, k-1$ , select  $s_i \leftarrow \mathbb{Z}_{q_i}$  and compute  $c_{i+1} = H_{i+1}(L, m, g_i^{s_i} y_i^{c_i} \bmod p_i)$ .
- D-3** (Forming the ring): Compute  $s_k = \alpha - x_k c_k \bmod q_k$ .

The resulting signature for  $m$  and  $L$  is  $(c_0, s_0, s_1, \dots, s_{n-1})$ .

#### [Signature Verification]

For  $i = 0, \dots, n-1$ , compute  $e_i = g_i^{s_i} y_i^{c_i} \bmod p_i$  and then  $c_{i+1} = H_{i+1}(L, m, e_i)$  if  $i \neq n-1$ . Accept if  $c_0 = H_0(L, m, e_{n-1})$ . Reject otherwise.

Intuitively, this scheme is a ring of the Schnorr signatures where each challenge is taken from the previous step. Indeed, it is the Schnorr signature scheme when  $n = 1$ .

**Theorem 2.** *The above all-DL scheme is unconditionally signer-ambiguous.*

*Proof.* Observe that all  $s_i$  are taken randomly from  $\mathbb{Z}_{q_i}$  except for  $s_k$  at the closing point. At the closing point,  $s_k$  also distributes uniformly over  $\mathbb{Z}_{q_k}$  since  $\alpha$  is uniformly chosen from  $\mathbb{Z}_{q_k}$ . Therefore, for fixed  $(L, m)$ ,  $(s_0, \dots, s_{n-1})$  has  $\prod_{i=0}^{n-1} q_i$  variation that are equally likely regardless of the closing point. Remaining  $c_0$  in a signature is determined uniquely from  $(L, m)$  and  $s_i$ 's.  $\square$

Note that the signer ambiguity does not rely on ideal randomness assumption on the hash function.

Next, we claim unforgeability. Let  $\mathcal{A}$  be a  $(\tau, \epsilon, q_s, q_h)$ -adversary that requests signing oracle at most  $q_s$  times and accesses random oracles at most  $q_h$  times in total and output forged  $(L, m, \sigma)$  with probability at least  $\epsilon$  and running time at most  $\tau$ . The following theorem can be proven (see Appendix B).

**Theorem 3.** *If there exists  $(\tau, \epsilon, q_s, q_h)$ -adversary  $\mathcal{A}$  for public-key set  $\mathcal{L}$  of size  $n$ , then there exists  $(\eta, \mu)$ -simulator  $\text{sim}$  that uses  $\mathcal{A}$  as a black-box and computes discrete-logarithm  $x_i$  of  $(y_i, p_i, q_i, g_i) \in \mathcal{L}$  for at least one  $i$  with probability at least  $\mu$  within running time  $\eta$ . Here,  $\eta < \frac{32q_h^2 + 4}{\epsilon} \cdot \tau$  and  $\mu > \frac{9}{100}$  under the condition that  $\epsilon > \frac{8q_h^2}{q}$  and  $q > 2q_hq_s$  where  $q$  is the smallest  $q_i$  included in  $\mathcal{L}$ .*

We remark that the running time only concerns the number of black-box execution of  $\mathcal{A}$ . Note also that the condition  $q > 2q_hq_s$  is not essential and used only for simplifying the presentation of the reduction cost so that the impact of each variable is comprehensible. If necessary, one can obtain detailed formula without the condition. These remarks apply to all the theorems in the rest of this paper.

## 5.2 All RSA Case

For  $i = 0, \dots, n-1$ , let  $(e_i, N_i)$  be RSA public-keys and  $H_i : \{0, 1\}^* \rightarrow \mathbb{Z}_{N_i}$  be hash functions. Let  $L$  be a list of these public-keys. A signer who has private key  $d_k$  generates a signature for message  $m$  as follows.

### [Signature Generation]

- T-1 (Initialization): Select  $r_k \leftarrow \mathbb{Z}_{N_k}$  and compute  $c_{k+1} = H_{k+1}(L, m, r_k)$ .
- T-2 (Forward sequence): For  $i = k+1, \dots, n-1, 0, \dots, k-1$ , select  $s_i \leftarrow \mathbb{Z}_{N_i}$ , and compute  $c_{i+1} = H_{i+1}(L, m, c_i + s_i^{e_i} \bmod N_i)$ .
- T-3 (Shaping into a ring): Compute  $s_k = (r_k - c_k)^{d_k} \bmod N_k$

The resulting signature for  $m$  and  $L$  is  $(c_0, s_0, s_1, \dots, s_{n-1})$ .

### [Signature Verification]

For  $i=0, \dots, n-1$ , compute  $r_i = c_i + s_i^{e_i} \bmod N_i$  and then  $c_{i+1} = H_{i+1}(L, m, r_i)$  if  $i \neq n-1$ . Accept if  $c_0 = H_0(L, m, r_{n-1})$ . Reject, otherwise.

Unconditional signer-ambiguity can be proven in the same way as that for the all DL-based scheme. Unforgeability is also proven in the similar way. We wrap random oracles for each hash function in the same way as done in the proof for the DL version. The following theorem is proven (see Appendix C).

**Theorem 4.** *If there exists  $(\tau, \epsilon, q_s, q_h)$ -adversary  $\mathcal{A}$  for public-key set  $\mathcal{L}$  of size  $n$ , then there exists  $(\eta, \mu)$ -simulator  $\text{sim}$  that, given  $(w_0, \dots, w_{n-1})$ , uses  $\mathcal{A}$  as a black-box, and computes  $w_i^{d_i} \bmod N_i$  for some  $i \in \{0, \dots, n-1\}$  with probability at least  $\mu$  and running-time within  $\eta$ . Here,  $\eta \approx \tau$  and  $\mu > \frac{1}{4q_h^2}\epsilon$  under the condition of  $N > 2q_h q_s$  where  $N$  is the smallest modulus among all  $N_i$  in  $\mathcal{L}$ .*

### 5.3 Mixture Case: RSA and Schnorr

We finally show a small example for involving both RSA and DL keys. For simplicity, we consider the case  $n = 2$ , i.e., only two public-keys are involved. Let  $L$  be consists of RSA public-key  $(e, N)$  and one Schnorr public-key  $(y, g, p, q)$ . Let  $H_0 : \{0, 1\}^* \rightarrow \mathbb{Z}_N$  and  $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be hash functions. A signer who has the RSA private key,  $d$ , generates a signature for message  $m$  as follows.

#### [Signature Generation]

**M-1** (Initialization): Select  $\beta \leftarrow \mathbb{Z}_N$  and compute  $c_1 = H_1(L, m, \beta)$ .

**M-2** (Forward sequence): Select  $s_1 \leftarrow \mathbb{Z}_q$  and compute  $c_0 = H_0(L, m, g^{s_1} y^{c_1} \bmod p)$ .

**M-3** (Shaping into a ring): Compute  $s_0 = (\beta - c_0)^d \bmod N$

The resulting signature is  $(c_0, s_0, s_1)$ .

#### [Signature Verification]

Given  $(L, m, c_0, s_0, s_1)$ , compute  $c_1 = H_1(L, m, c_0 + s_0^e \bmod N)$ . Accept if  $c_0 = H_0(L, m, g^{s_1} y^{c_1} \bmod p)$ . Reject, otherwise.

The signature can be shorten by selecting  $(c_1, s_1, s_0)$  as a signature because  $|c_0|$  is the size of RSA modulus typically  $> 1024$  bits while  $|c_1|$  is the size of  $q$  typically  $> 160$  bits.

Unconditional signer-ambiguity can be proven as well as the former examples. Regarding unforgeability, we prove the following theorem by following the similar way as done in the proof of Theorem 3 and 4. Sketch of the proof is shown in Appendix D.

**Theorem 5.** *The above scheme is existentially unforgeable against adaptive chosen message and chosen public-key attacks.*

## 6 Efficiency

We compare our ring signature scheme with the existing schemes using DL, ECDL(elliptic curve DL) and RSA trapdoor functions, in terms of the length of a signature and the computational cost of signature generation and verification. We refer the scheme in Section 3.1 by “WI signatures” and the scheme in Section 3.2 with RSA trapdoor function by “RSA ring signatures”, hereafter. Let  $n$  be the number of signers of ring signature.

Table 1 shows the comparison in terms of the length of signature. Here,  $L(\text{DL})$  is the length of exponent of DL signature, and is typically 160-bit.  $L(\text{RSA})$  is

**Table 1.** The table shows the length of signature and its typical value (bit).

	Length of signature	Typical value
WI signature	$(L(DL) + L(DL)) \times n$	$320 \times n$
Ours with DL	$L(DL) + L(DL) \times n$	$160 + 160 \times n$
Ours with ECDL	$L(EC) + L(EC) \times n$	$160 + 160 \times n$
RSA ring signature	$(L(RSA) + 160) + (L(RSA) + 160) \times n$	$1184 + 1184 \times n$
Ours with RSA	$L(RSA) + L(RSA) \times n$	$1024 + 1024 \times n$

**Table 2.** The table shows the computational costs of signature generation and verification and its typical value (arithmetic operation).

	Costs of generation	Typical value
WI signature	$T(DL) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with DL	$T(DL) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with ECDL	$T(EC) \times 5/4 \times n$	$7.1 \times 10^7 \times n$
RSA ring signature	$T(RSA^{-1}) + T(RSA) \times n$	$1.0 \times 10^9 + 1.6 \times 10^7 \times n$
Ours with RSA	$T(RSA^{-1}) + T(RSA) \times n$	$1.0 \times 10^9 + 1.6 \times 10^7 \times n$
	Costs of verification	Typical value
WI signature	$T(DL) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with DL	$T(DL) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with ECDL	$T(EC) \times 5/4 \times n$	$7.1 \times 10^7 \times n$
RSA ring signature	$T(RSA) \times n$	$1.6 \times 10^7 \times n$
Ours with RSA	$T(RSA) \times n$	$1.6 \times 10^7 \times n$

the length of modulus of RSA signature, and is typically 1024-bit.  $L(EC)$  is the length of the size of cyclic subgroup in elliptic curve, and is typically 160-bit. From the table, we can see that the length of our signature with DL is *one half* of WI signature for large  $n$ , and that the length of our signature with RSA is 0.8 of RSA ring signature.

Table 2 shows the comparison in terms of the computational costs of signature generation and verification. Here,  $T(DL)$ ,  $T(EC)$ ,  $T(RSA^{-1})$  and  $T(RSA)$  are the computational costs of modular exponentiation, scalar multiplication on elliptic curve, inverse RSA function and RSA function, respectively. Typically,  $T(DL) = T((1024)^{(160)})$ ,  $T(EC) = T((160) \cdot (EC160))$ ,  $T(RSA^{-1}) = T((1024)^{(1024)})$  and  $T(RSA) = T((1024)^{(16)})$ . Here,  $T((x)^{(y)})$  is the number of (single precision) arithmetic operation of exponentiation with  $x$ -bit modulus and  $y$ -bit exponent, and is estimated  $x^2 \times y$ . Exponentiation with  $y$ -bit exponent needs  $y$   $x$ -bit multiplications, using binary method and the fact costs of square is half of multiplication.  $x$ -bit multiplication needs  $x^2$  (single precision) arithmetic operations.  $T((y) \cdot (ECx))$  is the number of (single precision) arithmetic operation of scalar multiplication on elliptic curve with  $x$ -bit base field and  $y$ -bit scalar, and is estimated  $x^2 \times 14 \times y$ . Scalar multiplication on elliptic curve with

$y$ -bit scalar needs  $y$  additions of points, using binary method and the fact costs of doubling is half of costs of addition. Addition of points with  $x$ -bit base field needs  $14$   $x$ -bit multiplications, using Jacobian coordinate.  $x$ -bit multiplication needs  $x^2$  (single precision) arithmetic operations. Hence, we have

$$\begin{aligned} T((1024)^{(1024)}) &= 1024^2 \times 1024 \approx 1.07 \times 10^9, \\ T((1024)^{(160)}) &= 1024^2 \times 160 \approx 1.67 \times 10^8, \\ T((1024)^{(16)}) &= 1024^2 \times 16 \approx 1.67 \times 10^7, \\ T((160) \cdot (EC160)) &= 160^2 \times 14 \times 160 \approx 5.73 \times 10^7. \end{aligned}$$

The computational costs of exponentiation with two basis is  $5/4$  of exponentiation with single basis, using two basis binary method. From the table, we can see that the computational costs of our signature with DL is as same as WI signature, and that the computational costs of our signature with RSA is as same as RSA ring signature.

Notice that in known schemes the length and the computational costs of signature is proportional to the *maximum* of the length of DL exponent / RSA modulus. In our scheme, the length and the computational costs of signature is proportional to the *average* of the length of DL exponent / RSA modulus, since our scheme need not to round up the length to the maximum length.

## Acknowledgements

The authors are grateful to Emmanuel Bresson and the anonymous referees for their helpful comments.

## References

1. M. Abe and F. Hoshino. Remarks on mix-network based on permutation network. *PKC 2001*, LNCS 1992, pp. 317–324. Springer-Verlag, 2001.
2. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Asiacrypt 2001*, LNCS 2248, pp. 514–532. Springer-Verlag, 2001.
3. M. Bellare, and P. Rogaway. Random Oracles are practical: a paradigm for designing efficient protocols. *1st ACM CCCS*, pp. 62–73. ACM, 1993.
4. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. *CRYPTO 2002*, LNCS 2442, pp. 465–480. Springer-Verlag, 2002.
5. J. Camenisch. Efficient and generalized group signatures. *EUROCRYPT '97*, LNCS 1233, pp. 465–479. Springer-Verlag, 1997.
6. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. *EUROCRYPT '99*, LNCS 1592, pp. 107–122. Springer-Verlag, 1999.
7. C. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. *EUROCRYPT '98*, LNCS 1403, pp. 561–575. Springer-Verlag, 1998.
8. D. Chaum and E. Van Heyst. Group signatures. *EUROCRYPT '91*, LNCS 547, pp. 257–265. Springer-Verlag, 1991.



9. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *CRYPTO '94*, LNCS 839, pp. 174–187. Springer-Verlag, 1994.
10. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. *EUROCRYPT '96*, LNCS 1070, pp. 72–83. Springer-Verlag, 1996.
11. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *EUROCRYPT '97*, LNCS 1233, pp. 103–118. Springer-Verlag, 1997.
12. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1:77–94, 1988.
13. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. STOC'90, pp. 416–426, 1990.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO '86*, LNCS 263, pp. 186–199. Springer-Verlag, 1987.
15. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.
16. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. *EUROCRYPT '88*, LNCS 330 of *Lecture Notes in Computer Science*, pp. 123–128. Springer-Verlag, 1988.
17. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. *EUROCRYPT '96*, LNCS 1070, pp. 143–154. Springer-Verlag, 1996.
18. M. Naor. Deniable ring authentication. *CRYPTO 2002*, LNCS 2442, pp. 481–498. Springer-Verlag, 2002.
19. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. *CRYPTO '98*, LNCS 1462, pp. 354–369. Springer-Verlag, 1998.
20. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 2000.
21. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *Asiacrypt 2001*, LNCS 2248, pp. 552–565. Springer-Verlag, 2001.
22. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. *FOCS'94*, pp. 454–465, 1994.
23. C. P. Schnorr. Efficient signature generation for smart cards. *J. Cryptology*, 4(3):239–252, 1991.

## Appendix A

The following an efficient 1-out-of-n signature scheme in non-separable model based on the representation problem.

Let  $p, q$  be large primes. Let  $\langle g \rangle$  denote a prime subgroup in  $\mathbb{Z}_p^*$  generated by  $g$  whose order is  $q$ . Let  $x_i, y_i$  be  $y_i = g^{x_i} \bmod p$ . Here  $x_i$  is the secret-key and  $(y_i, p, q, g)$  is the public-key. All member use common  $p, q, g$  in the non-separable model. So only  $y_i$  is different in public-keys for each member. Let  $L$  be a set of  $(y_i, p, q, g)$  for  $i = 1, \dots, n$ . Let  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be a hash function.

A signer who owns secret key  $x_k$  generates a signature for message  $m$  with public-key list  $L$  that includes  $y_k$ , in the following way.

**S-1** Select  $\alpha, c_i \leftarrow \mathbb{Z}_q$  for  $i = 0, \dots, n-1$ ,  $i \neq k$ , and compute  $z = g^\alpha \prod_{i=0, i \neq k}^{n-1} y_i^{c_i} \bmod p$ .

**S-2** Compute

$$c = H(L, m, z)$$

$$c_k = c - (c_0 + \dots + c_{k-1} + c_{k+1} + \dots + c_{n-1}) \bmod q$$

$$s = \alpha - c_k \cdot x_k \bmod q.$$

The resulting signature is  $\sigma = (s, c_0, \dots, c_{n-1})$ .  $(L, m, \sigma)$  is valid if

$$\sum_{i=0}^{n-1} c_i \equiv H(L, m, g^s y_0^{c_0} \dots y_{n-1}^{c_{n-1}}) \bmod q.$$

The security of this scheme can also be reduced to the discrete-log problem by rewinding simulation. But the reduction is quite costly because we may have to have at most  $n$  successful rewinding simulations to extract only one secret-key (in this worst case, all the secret-keys are extracted at once).

## Appendix B (Proof of Theorem [3](#))

To get the black-box  $\mathcal{A}$  run properly,  $\text{sim}$  simulates the random oracles that corresponds to each hash function and the signing oracle. For simplicity, the random oracles are treated as a single oracle that takes  $Q_j = (i, L_j, m_j, r_j)$  as  $j$ -th query and returns a random value that corresponds to  $H_i(L_j, m_j, r_j)$  maintaining consistency against duplicated queries. The signing oracle receives the  $j$ -th query, say  $R_j = (L_j, m_j)$ , to sign. To avoid complicated suffixes, we describe a public-key with a suffix relative to  $L_j$  in the current context. So,  $y_0 \in L_j$  and  $y_0 \in L_{j'}$  could differ. We hope that this should cause no confusion. The corresponding answer is simulated in the following way.

**D'-1:** Choose  $c_0 \leftarrow \mathbb{Z}_{q_0}$ .

**D'-2:** For  $i = 0, \dots, |L_j| - 1$ , select  $s_i \leftarrow \mathbb{Z}_{q_i}$ , compute  $e_i = g_i^{s_i} y_i^{c_i} \bmod p_i$ , and then compute  $c_{i+1} = H_{i+1}(L_j, m_j, e_i)$  if  $i \neq |L_j| - 1$ .

**D'-3:** Assign  $c_0$  to the value of  $H_0(L_j, m_j, e_{|L_j|-1})$ .

The simulation fails if Step D'-3 causes inconsistency in  $H_0$ . It happens with probability at most  $q_h/q$  where  $q$  is the smallest  $q_i$  in  $\mathcal{L}$ . Hence, the simulation is successful  $q_s$  times with probability at least  $(1 - q_h/q)^{q_s} \geq 1 - q_h q_s / q$ .

Let  $\Theta, \Omega$  be the random tapes given to the signing oracle and  $\mathcal{A}$ . The success probability of  $\mathcal{A}$  is taken over the space defined by  $\Theta, \Omega$  and random oracle  $H$ . Let  $\mathcal{S}$  be a set of  $(\Theta, \Omega, H)$  with which  $\mathcal{A}$  is successful in forgery. From the definition of  $\epsilon$ , we have  $\Pr[(\Theta, \Omega, H) \in \mathcal{S}] \geq \epsilon$ . Let  $(L, m, c_0, s_0, \dots, s_{n'-1})$  be a forged signature  $\mathcal{A}$  outputs. Here  $n' = |L|$ . Define  $r_i = g_i^{s_i} y_i^{c_i} \bmod p_i$  and  $c_{i+1} = H_{i+1}(L, m, r_i)$  for  $i = 0, \dots, n' - 1$  (indices are taken modulo  $n'$ ). Then, with probability at least  $1 - 1/q$ , there exist queries  $Q_j = (i + 1, L, m, r_i)$  for all  $i = 0, \dots, n' - 1$  due to the ideal randomness of  $H$ . Let  $\mathcal{S}'$  be a subset of  $\mathcal{S}$  where  $(\Theta, \Omega, H) \in \mathcal{S}'$  leads  $\mathcal{A}$  to output a signature that has corresponding queries

as above with successfully simulated signing oracle. Then,  $\Pr[(\Theta, \Omega, H) \in \mathcal{S}'] \geq (1 - q_h q_s / q)(1 - 1/q)\epsilon$ . Let  $\epsilon' = (1 - q_h q_s / q)(1 - 1/q)\epsilon$ .

Since the queries form a ring, there exists at least one index, say  $k$ , in  $\{0, \dots, n' - 1\}$  such that  $Q_u = (k + 1, L, m, r_k)$  and  $Q_v = (k, L, m, r_{k-1})$  satisfy  $u \leq v$ . Namely,  $k$  is in between the gap of query order. We call such  $(u, v)$  a gap index. Note that  $u = v$  happens only if  $n' = 1$ , which means that resulting  $L$  contains only one public-key. If there are two or more gap indices with regard to a signature, only the smallest one is considered. We classify  $\mathcal{S}'$  by the gap indices. Let  $\mathcal{S}'_{u,v}$  denote a class where  $(\Theta, \Omega, H) \in \mathcal{S}'_{u,v}$  yields gap indices  $(u, v)$ . There are at most  $\binom{2}{q_h} + \binom{1}{q_h} = q_h(q_h + 1)/2$  classes. By invoking  $\mathcal{A}$  with randomly chosen  $(\Theta, \Omega, H)$  at most  $t_1 = 1/\epsilon'$  times,  $\text{sim}$  finds at least one  $(\Theta, \Omega, H) \in \mathcal{S}'_{u,v}$  for some gap index  $(u, v)$  with probability  $1 - \exp(-1) > 3/5$ .

We consider a set of gap indices that is likely to appear when  $(\Theta, \Omega, H)$  is chosen randomly. Let  $GI = \{(u, v) \mid |\mathcal{S}'_{u,v}|/|\mathcal{S}'| \geq \frac{1}{q_h(q_h+1)}\}$  and  $B = \{(\Theta, \Omega, H) \in \mathcal{S}'_{u,v} \mid (u, v) \in GI\}$ . Then, it holds that  $\Pr[B|\mathcal{S}'] \geq \frac{1}{2}$ . Due to this fact known as heavy-row lemma,  $(\Theta, \Omega, H)$  that yields the successful run of  $\mathcal{A}$  is in  $B$  with probability at least  $1/2$ .

Split  $H$  as  $(H^-, c_k)$  where  $H^-$  corresponds to the answers to all queries except for  $Q_v$  answered with  $c_k$ . Due to the heavy-row lemma, again, with probability at least  $1/2$ ,  $(\Theta, \Omega, H^-)$  satisfies  $\Pr_{c'_k}[(\Theta, \Omega, H^-, c'_k) \in \mathcal{S}'_{u,v}] \geq \frac{\epsilon'}{2q_h(q_h+1)}$ . Since we assume  $\epsilon > \frac{8q_h^2}{q}$  and  $q > 2q_h q_s$ , it holds that  $\frac{\epsilon'}{2q_h(q_h+1)} > 1/q$ .

By running  $\mathcal{A}$  up to  $t_2 = (\frac{\epsilon'}{2q_h(q_h+1)} - \frac{1}{q})^{-1}$  times with  $(\Theta, \Omega, H^-)$  obtained in the first successful run and randomly chosen  $c'_k (\neq c_k)$ , then, with probability at least  $3/5$ ,  $\text{sim}$  finds at least one  $c'_k$  such that  $(\Theta, \Omega, H^-, c'_k) \in \mathcal{S}_{u,v}$ . Since  $Q_u$  happens before  $Q_v$ ,  $r_i$  is unchanged for both runs. Therefore,  $\text{sim}$  can compute the discrete-log,  $x_k = (s_k - s'_k)/(c'_k - c_k) \bmod q_k$ . Overall success probability is

$$\mu > \frac{3}{5} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{3}{5} = \frac{9}{100},$$

and the number of invocation of  $\mathcal{A}$  is

$$t_1 + t_2 < \frac{1}{\epsilon'} + \frac{4q_h(q_h+1)}{\epsilon'} < \frac{4}{\epsilon} + \frac{4 \cdot 4 \cdot 2q_h^2}{\epsilon} = \frac{32q_h^2 + 4}{\epsilon}.$$

## Appendix C (Proof of Theorem [4](#))

The first half of the proof is the same as the one for Theorem [3](#) (the simulation of the signing oracle is different but can be done only with trivial changes). That is, there exists class  $\mathcal{S}'$  such that  $(\Theta, \Omega, H) \in \mathcal{S}'$  results in a successful simulation of the signing oracle and the forged signature has corresponding queries to the random oracle. Accordingly,  $\Pr[(\Theta, \Omega, H) \in \mathcal{S}'] \geq (1 - q_h q_s / N)(1 - 1/N)\epsilon$ .

At the beginning of the simulation,  $\text{sim}$  selects a pair of index  $(u, v)$  randomly so that  $1 \leq u \leq v \leq q_h$ . With probability  $2/q_h(q_h + 1)$ , the guess is correct and  $\text{sim}$  receives  $Q_u = (k + 1, L, m, r_k)$  and  $Q_v = (k, L, m, r_{k-1})$  so that  $(u, v)$  is a gap

index. Let  $k'$  be an index such that  $vk_{k'} \in \mathcal{L}$  corresponds to  $vk_k \in L$ . When query  $Q_v$  is made ( $u$ -th query has been already made by this moment), **sim** returns  $c_k = r_k - w_{k'} \bmod N_k$  as a value of  $H_k(L, m, r_{k-1})$ . If  $\mathcal{A}$  is successful in forgery, it outputs  $s_k$  that satisfies  $r_k \equiv c_k + s_k^{e_k} \bmod N_k$ . Since  $r_k \equiv c_k + w_{k'} \bmod N_k$ , we obtain  $s_k$  as the inverse of  $w_{k'}$  with regard to the public-key  $vk_{k'}$  in  $\mathcal{L}$ .

Overall success probability of **sim** is

$$\mu \geq \frac{(1 - q_h q_s / N)(1 - 1/N)}{q_h(q_h + 1)/2} \epsilon > \frac{1}{4q_h^2} \epsilon.$$

The rightmost term assumes  $N > 2q_h q_s$ . The running time  $\eta$  is almost the same as  $\tau$  as **sim** runs  $\mathcal{A}$  only once and the simulation cost for the signing oracle and the random oracle are assumed to be sufficiently smaller than  $\tau$ .

## Appendix D (Proof of Theorem 5)

We show that if there exists  $(\tau, \epsilon, q_s, q_h)$ -adversary  $\mathcal{A}$  for a set of public-key,  $L$ , of size  $n$ , then there exists  $(\tau', \epsilon')$ -simulator **sim** such that using  $\mathcal{A}$  as a black-box, it computes  $w^d \bmod N$  for  $w \leftarrow N$  with probability greater than  $3/5$  and running time no more than  $\frac{4q_h^2}{\epsilon} \tau$ , or computes the discrete-logarithm of  $y$  with probability greater than  $\frac{9}{100}$  and running time no more than  $\frac{32q_h^2 + 4}{\epsilon} \tau$ .

Let  $\gamma$  be  $\min(q, N)$ . We assume that  $\gamma > 2q_h q_s$  and  $\epsilon > \frac{8q_h^2}{q}$ .

The proof is by combining the proofs for Theorem 3 and 4. Let  $\epsilon' = (1 - q_h q_s / |\gamma|)(1 - 1/|\gamma|)\epsilon$ .

Simulator **sim** guesses  $(u, v)$  and runs  $\mathcal{A}$ . If  $Q_v = (0, L, m, r_1)$  for some  $L, m$  and  $r_1$ , **sim** checks if  $Q_u = (1, L, m, r_0)$  for the same  $L, m$  and some  $r_0$ . If it is the case, **sim** chooses  $t \leftarrow N$  and returns  $c_0 = r_0 - wt^e \bmod N$  as the value of  $H_0(L, m, r_1)$ . If  $\mathcal{A}$  succeeds and  $(u, v)$  forms a gap, it holds that  $s_k = (r_0 - c_0)^d \equiv (wt^e)^d \equiv w^d t \bmod N$ . Accordingly,  $s_k/t = w^d \bmod N$ . In this case, simulation ends here successfully. Such a successful case happens with probability greater than  $3/5$  while repeating the simulation at most  $\left\{ \frac{(1 - q_h q_s / N)(1 - 1/N)}{q_h(q_h + 1)/2} \epsilon \right\}^{-1} < \frac{4q_h^2}{\epsilon}$  (this is straightforward from the proof of Theorem 4 in Appendix C).

For all other cases, **sim** proceeds as follows. Regardless of the initial guess of  $(u, v)$ , **sim** completes an execution of  $\mathcal{A}$ . If  $\mathcal{A}$  succeeds and the resulting gap index, say  $(u', v')$ , corresponds to the queries  $Q_{u'} = (0, L, m, r_1)$  and  $Q_{v'} = (1, L, m, r_0)$ , namely, if the resulting gap index comes across the discrete-log key, **sim** proceeds to rewinding simulation with the forking point  $v'$  in the same way as done in the proof of Theorem 3. As a result, **sim** gets a collision and computes the discrete-log,  $x$ . The success probability and the running time for this case is the same as that in the proof of Theorem 3.

# A Revocation Scheme with Minimal Storage at Receivers

Tomoyuki Asano\*

Sony Corporation, 6-7-35 Kitashinagawa, Shinagawa-ku, Tokyo 141-0001, Japan,  
tomo@arch.sony.co.jp

**Abstract.** A revocation or a broadcast encryption technology allows a sender to transmit information securely over a broadcast channel to a select group of receivers excluding some revoked receivers. In this paper we propose two efficient revocation methods which are suitable for stateless receivers. The proposed methods use an  $a$ -ary key tree structure and require at most  $r \left( \frac{\log(N/r)}{\log a} + 1 \right)$  ciphertexts broadcast. Our Method 1 requires only one key to be stored and  $O \left( \frac{2^a \log^5 N}{\log a} \right)$  computational overhead at a receiver, whereas Method 2 requires  $\frac{\log N}{\log a}$  keys and  $O(2^a)$  computational overhead, where  $N$  and  $r$  respectively denote the total number of receivers and the number of revoked receivers. Our methods are very efficient with respect to the number of keys each receiver stores, especially Method 1 minimizes it.

## 1 Introduction

Recent advances in technology give us a lot of ways to distribute digital data without loss of quality. We can easily use, modify and exchange many kinds of digital data such as digital pictures or music. However, those advances have caused serious challenges related to *copyright protection* or *digital rights management* issues. Though copyright-protected data (e. g. music, movies or TV programs) should be treated under conditions to which its copyright holder agrees, various kinds of such content can be recorded, copied or exchanged in an illegal manner. One of the technologies that are being used to protect such data is called *revocation scheme* or *broadcast encryption scheme*. This technology allows a sender to transmit information securely over a broadcast channel to a select group of receivers. The sender may exclude some receivers (called *revoked receivers*) and enable only legitimate receivers to obtain the transmitted information.

Revocation schemes are used in many real world applications. For example, in a pay-TV system, users can watch TV programs if they subscribe to the service and pay the fee. If some users do not pay for the programs, they might be excluded, so that they will not be able to watch the program the following month even if they own an appropriate receiver. Other examples are CPPM

---

\* Most of this work was done while the author was visiting Stanford University.

and CPRM [8] which are systems protecting copyrighted content stored on pre-recorded or recordable media from unauthorized copying. In these systems only compliant receivers (i.e. players or recorders) that are manufactured under a certain license contract can retrieve secret information (called *session key*) from a medium using their receiver keys. The session key is required for decryption or encryption of the content file stored on the medium. If it is found that there is a receiver which does not obey the license contract this receiver will be revoked, and as a result it will no longer be able to retrieve the session keys distributed after the revocation.

For general receivers (e.g. consumer electronics devices), the easiest way to store secret information such as receiver keys is storing it as part of the initial configuration at manufacturing time. Giving a mechanism to receivers for changing keys they store increases the production cost and might also weaken their security. Therefore it is preferable in most cases to assume that receivers can not change their keys. Such receivers are called *stateless receivers*. As described in [17], typical examples of stateless receivers are off-line devices, such as CD or DVD players. In this paper we propose two efficient revocation methods that are suitable for stateless receivers.

The organization of this paper is as follows. We introduce related work and contributions of this paper in the rest of this section. Section 2 describes two revocation methods proposed in this paper. We discuss the security of our methods in section 3, some techniques and the properties of those methods in section 4. We present a modification of CPPM and CPRM in Section 5. Our results in this paper are summarized in section 6.

## 1.1 Related Work

As described in the previous section, a revocation scheme or broadcast encryption scheme allows a sender to transmit information securely over a broadcast channel to a select group of receivers. Let  $N$  and  $r$  be the total number of receivers in the system and the number of revoked receivers, respectively. A naive method to implement this scheme is as follows. Assume each receiver owns a unique key. A sender broadcasts secret information encrypted under each of the unique keys owned by the non-revoked receivers. This method requires each receiver to store only one key, but the sender must transmit  $N - r$  ciphertexts. Since a large amount of bandwidth is necessary for large  $N$ , this method is not suitable for applications where the bandwidth for such data is restricted.

There exists another naive method where the size of the broadcast message is minimized. We call it the Power Set Method. The method defines a power set of  $N$  receivers, i.e.  $\{S_{b_1 b_2 \dots b_i \dots b_N}\}$  where  $b_i \in \{0, 1\}$ . Each  $b_i$  indicates whether or not a receiver  $i$  belongs to a subset  $S_{b_1 b_2 \dots b_i \dots b_N}$ . It assigns a subset key for each subset and gives the subset key to receivers which belong to the subset. To send secret information to an arbitrary group of receivers, a sender chooses a subset where  $b_i = 1$  only for selected receivers  $i$ , encrypts the information with a subset key corresponding to the subset, and broadcasts the ciphertext. This method requires the sender to broadcast only one ciphertext, while each

receiver needs to store  $2^{N-1}$  keys. Hence this method is not suitable for receivers in many applications if  $N$  is large. However, we use this technique in conjunction with a key tree structure in order to reduce the number of ciphertexts which are broadcast in our methods.

The notion of *broadcast encryption* was introduced by Berkovits [3] and Fiat et al. [10] independently. The main criteria for this technology are the number of ciphertexts (the length of the message) to be broadcast, the number of keys each receiver stores, and the computational overhead at a receiver. Berkovits constructed a broadcast encryption method using a secret sharing scheme [22]. This method requires each receiver to store one key, however the length of the broadcast message is  $O(N)$ . Fiat et al. proposed an  $r$ -resilient method which is resistant to a collusion of up to  $r$  revoked receivers by combining their 1-resilient methods hierarchically. This method requires message length of  $O(r^2 \log^2 r \log N)$  and the storage of  $O(r \log r \log N)$  at each receiver.

Wallner et al. [24] and Wong et al. [25] independently proposed efficient methods using a logical key tree structure. Their methods define a logical tree and a node key for each node of the tree. Each receiver is assigned to a leaf of the tree and given a set of node keys defined for the nodes on the path from the leaf to the root of the tree. Therefore, each receiver stores  $\log N + 1$  keys, assuming that the system uses a binary tree. All of these keys except one are shared by other receivers. This method revokes one receiver at a time, and updates all keys stored by non-revoked receivers, which have also been owned by the revoked receiver. A sender needs to broadcast  $2 \log N$  ciphertexts and a receiver needs to perform at most  $\log N$  decryptions for this single revocation. If the system needs to revoke  $r$  receivers by repeating the single revocation, the sender has to send  $2r \log N$  ciphertexts.

Since the key tree structure has good properties, modifications of the methods of [24,25] have been proposed [5,11,16,17]. Some of them reduce the messages for a single revocation to  $\log N$  by combining the key tree structure with another technique. McGrew et al. [16] used a one-way function, Canetti et al. [5] used a pseudo random generator, and Kim et al. [11] used Diffie-Hellman key exchange scheme [9]. The number of keys a receiver stores remains  $\log N + 1$ , while their methods increase the computational overhead at a receiver, namely, each receiver needs to perform the computation of such a technique at most  $\log N$  times. Similar to their original methods, they assume non-stateless receivers, i. e. receivers have a capability to change their keys.

If receivers are not stateless, they can store keys (e. g. shared keys established among the sender and a group of receivers) given at time  $t_1$  and use them at time  $t_2$  (where  $t_1 < t_2$ ) to obtain the current session key. This may contribute to reduce the size of the broadcast. On the other hand, stateless receivers can store only the keys given at the initial stage such as manufacturing time. Hence every broadcast message must contain enough information to enable non-revoked receivers to obtain the current session key using their initial receiver keys.

Kumar et al. [13] proposed revocation methods using error correcting codes. In their methods only non-revoked receivers can correct the error in the broad-

cast message and retrieve the secret information. Their construction based on polynomials requires messages of  $O(r \log N)$  broadcast and storage of  $O(r \log N)$  at a receiver, and their construction based on algebraic-geometric codes requires message of  $O(r^2)$  and  $O(r \log N)$  storage overhead. The latter construction is interesting because the length of the message is independent of the number of total receivers.

Anzai et al. [2] and Naor et al. [18] independently proposed other methods using a secret sharing scheme. The main advantage of their methods is the size of storage at receivers. Their methods require receivers to store an element in a certain group. On the other hand, the methods require  $O(w)$  messages broadcast and  $O(w)$  exponentiations performed at a receiver, where  $w$  is the upper bound of the number of revoked receivers in the system which is fixed in advance. In other words, if we set the system resistant to a collusion of any number of revoked receivers then  $O(N)$  messages and  $O(N)$  exponentiations are required, regardless of the number of receivers actually revoked. Matsuzaki et al. [15] modified their method to reduce the computational overhead at a receiver to two modular exponentiations.

CPPM and CPRM [8] are methods for protection of copyrighted content stored on pre-recorded or recordable media (e.g. disks or semiconductor memories) that work with stateless receivers. Those methods require a prefixed number of ciphertexts being broadcast on the media and a relatively small number of keys being stored at a receiver. However, their revocation capability is restricted. We present detailed explanation of their properties as well as a modification of them in section 5.

Naor et al. [17] proposed two efficient methods suitable for stateless receivers using a binary key tree construction. The Complete Subset Method requires a sender to broadcast  $r \log(N/r)$  ciphertexts and each receiver to store  $\log N + 1$  keys, whereas the Subset Difference Method using a pseudo random sequence generator requires  $2r - 1$  ciphertexts,  $\frac{1}{2} \log^2 N + \frac{1}{2} \log N + 1$  keys and  $O(\log N)$  computational overhead at a receiver.

Luby et al. [14] and Poovendran et al. [20] analyzed the criteria of broadcast encryption schemes under information theoretic concepts. Since the methods we propose in this paper are constructed upon a computational assumption, their bounds are not applicable to them.

Our methods use a key tree structure and are suitable for stateless receivers. They provide a good balance in the criteria for the revocation technology, and are more efficient with respect to the number of keys stored at each receiver compared to previously proposed methods with such a structure. Especially one of our methods requires receivers to store only one key.

Another topic related to a revocation technology is a *traitor tracing* technology introduced by Chor et al. [7]. This is used to find a receiver who contributed for production of a non-legitimate receiver device or software by giving its secret information (e.g. receiver keys). Many schemes with traitor tracing capability, such as [4,7,17,18], have been proposed. We briefly explain the applicability of our methods to a traitor tracing scheme in section 4.4.



## 1.2 Our Results

In this paper we propose two efficient revocation methods suitable for stateless receivers. The Master Key technique due to Chick et al. [6] (a similar technique is also described in [10]) contributes to reduce the number of keys each receiver stores, and the Power Set Method used in conjunction with an  $a$ -ary logical key tree structure helps to reduce the number of ciphertexts to be broadcast, where the parameter  $a$  can be any positive integer satisfying  $a > 1$ . In turn, our methods require receivers to perform some computations.

The properties of our methods are shown in Table 1. For comparison, this table also contains the properties of the Complete Subset Method (CSM) and the Subset Difference Method (SDM) proposed in [17], which are considered to be most efficient among the methods proposed previously. In Method 1 we construct a revocation method which requires receivers to store only one key (a master key). Therefore, this method achieves minimal storage overhead for receivers. Method 2 is a variant of Method 1 which reduces the computational cost incurred by receivers to derive a key used for decryption of broadcast ciphertext from their master key in exchange for an increase in the number of master keys they store.

Table 1 tells that although our methods require more computational overhead of receivers, they are more efficient than other methods with respect to the number of keys each receiver stores. Our methods are also more efficient than CSM with regard to the number of ciphertexts broadcast. Since the Master Key technique is based on the security of RSA cryptosystem [21], the size of each master key in our methods is the size of a secure RSA modulus. Note that as analyzed in [17] a receiver in each method in Table 1 needs  $O(\log \log N)$  lookup operations (in order to find an appropriate ciphertext to decrypt) and a single decryption operation which are omitted from the table.

We show that our methods are secure under the assumption related to the RSA cryptosystem. Then we discuss some techniques which are used in our methods to reduce the size of the broadcast and the size of the storage at a receiver. We also provide a modification of CPPM and CPRM using the Master Key technique which reduces the size of the storage at receivers.

**Table 1.** The properties of methods in [17] and our methods

	CSM [17]	SDM [17]	Method 1	Method 2
Number of ciphertexts	$r \log(N/r)$	$2r - 1$	$r \left( \frac{\log(N/r)}{\log a} + 1 \right)$	$r \left( \frac{\log(N/r)}{\log a} + 1 \right)$
Number of keys @ receiver	$\log N$	$\frac{1}{2} \log^2 N$	1	$\frac{\log N}{\log a}$
Comp. cost for key derivation				
Pseudo-random generator	—	$O(\log N)$	—	—
Generation of primes	—	—	$O\left(\frac{2^a \log^5 N}{\log a}\right)$	—
Num. of multiplications	—	—	$\frac{(2^{a-1} - 1) \log N}{\log a}$	$2^{a-1} - 1$
Num. of modular exp.	—	—	1	1

## 2 The Proposed Methods

In this section we introduce two efficient revocation methods. These methods use the Power Set Method' defined below as an elemental technique.

**Power Set Method' with  $n$  Elements.** *Suppose there is a set of  $n$  elements  $i$  ( $i = 1, \dots, n$ ). Define  $2^n - 2$  subsets  $S_{b_1 b_2 \dots b_i \dots b_n}$  where  $b_i \in \{0, 1\}$ ,  $\sum_{i=1}^n b_i \neq 0$  and  $\sum_{i=1}^n b_i \neq n$  (which are elements of a power set except all  $b_i$ 's are 0 and 1). Assign a subset key for each subset. Give subset keys corresponding to subsets where  $b_i = 1$  to an element  $i$ . To send secret information to an arbitrary group of elements (except a group which consists of all elements), choose a subset where  $b_i = 1$  only for selected elements  $i$ , encrypt the information with a subset key corresponding to the subset and send the encrypted information.*

Assume  $N$  is a power of a positive integer  $a$ . Our methods adopt a logical  $a$ -ary tree called the Hierarchical Key Tree (HKT) and the Power Set Method' with  $a$  elements for each internal node (including the Root<sup>1</sup>) in the HKT. Basically  $2^a - 2$  subsets are defined for each internal node. A subset key is chosen for each subset and  $2^{a-1} - 1$  subset keys are given to each child node of the internal node. A receiver is assigned to a leaf of the HKT. Let  $path_j$  be a path from the leaf to which a receiver  $u_j$  is assigned to the Root. A receiver  $u_j$  is given master key(s) that can derive any subset key given to a node on  $path_j$ . Transmission of secret information including revocation of receivers is performed by broadcasting one or more ciphertexts encrypted under a subset key. This construction has a good property such that only one ciphertext needs to be sent for secure transmission to an arbitrary set of child nodes of a certain internal node in the HKT.

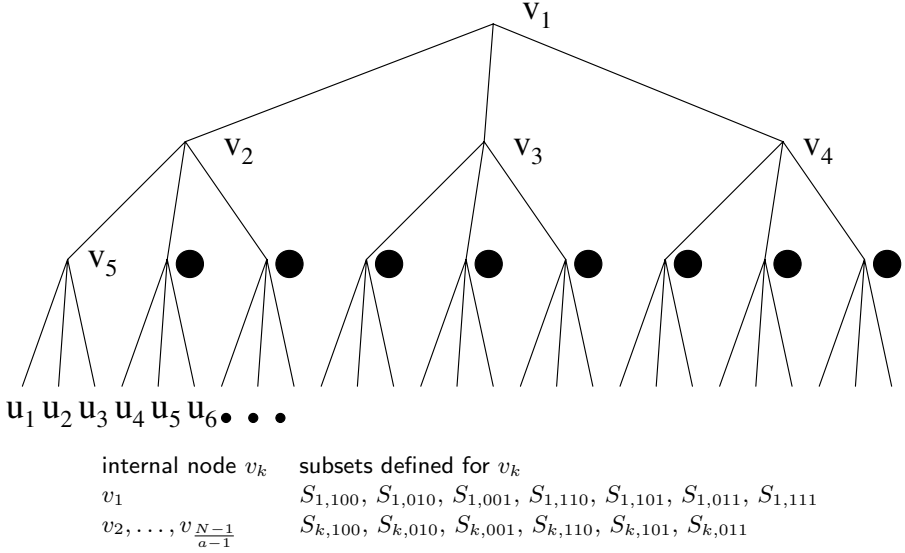
We have two ways of applying the Master Key technique to a revocation scheme. In Method 1 we adopt the Master Key system for the whole HKT. A receiver  $u_j$  is given a master key of  $(2^{a-1} - 1) \log_a N + 1$  subset keys corresponding to the subsets to which the receiver belongs, i. e. those subset keys are given to the nodes on  $path_j$ . Note that those subsets contain a subset to which all of  $N$  receivers belong, and the corresponding subset key is used if no receivers are revoked. In Method 2 we apply the Master Key system to each internal node in the HKT. In this method  $\log_a N$  master keys are given to a receiver  $u_j$ . Each master key can derive at most  $2^{a-1}$  subset keys corresponding to subsets defined for a node on  $path_j$ , to which the receiver  $u_j$  belongs.

### 2.1 Method 1

#### Setup

*Step 1.* Trusted Center (TC) which is a sender of secret information defines a rooted full  $a$ -ary HKT with  $N$  leaves. Each internal node in the HKT is named  $v_k$  ( $k = 1, \dots, \frac{N-1}{a-1}$ ) where the Root is  $v_1$  and other nodes are named with

<sup>1</sup> For clarity, we write the root of the HKT as 'the Root'.



**Fig. 1.** Subsets defined for internal nodes of the tree

breadth first order. A receiver  $u_j$  ( $j = 1, \dots, N$ ) is assigned to each leaf of the HKT. TC defines  $2^a - 2$  subsets  $S_{k,b_1b_2\dots b_i\dots b_a}$  where  $b_i \in \{0, 1\}$ ,  $\sum_{i=1}^a b_i \neq 0$  and  $\sum_{i=1}^a b_i \neq a$  for an internal node  $v_k$ . TC also defines a subset  $S_{1,11\dots 1}$  for the Root. TC selects large primes  $q_1$  and  $q_2$  and publishes  $M (= q_1 q_2)$ .

Figure 1 shows an example of the HKT for  $a = 3$  and  $N = 27$ , and subsets which are defined for internal nodes. The way to define the subsets is common to both of Method 1 and Method 2.

*Step 2.* TC chooses  $(2^a - 2) \frac{N-1}{a-1} + 1$  primes  $p_{k,b_1b_2\dots b_i\dots b_a}$  where  $k = 1, \dots, \frac{N-1}{a-1}$ ,  $b_i \in \{0, 1\}$ ,  $\sum_{i=1}^a b_i \neq 0$  for all  $k$  and  $\sum_{i=1}^a b_i \neq a$  for  $k \neq 1$ . Let  $B$  denote  $b_1b_2\dots b_i\dots b_a$ . Then TC assigns  $p_{k,B}$  to a subset  $S_{k,B}$  and publishes this assignment. Let  $T$  be a product of all primes assigned to the subsets. TC randomly chooses an element  $K \in \mathbb{Z}_M^*$  and sets a subset key  $SK_{k,B}$  corresponding to a subset  $S_{k,B}$  as

$$SK_{k,B} = K^{T/p_{k,B}} \bmod M$$

TC (imaginarily) gives subset keys  $SK_{k,B}$  with  $b_i = 1$  to  $i^{\text{th}}$  child node of the internal node  $v_k$ . Therefore,  $2^{a-1} - 1$  subset keys are given to each of the child nodes of an internal node. In addition, a subset key  $SK_{1,11\dots 1}$  is given to each of the child nodes of the Root.

*Step 3.* TC gives a receiver  $u_j$  a master key  $MK_j$  of  $(2^{a-1} - 1) \log_a N + 1$  subset keys that are given to the nodes on  $path_j$ . Let  $w_j$  be a product of all primes assigned to the subsets to which the receiver  $u_j$  belongs (i.e. the corresponding

subset keys are given to the nodes on  $path_j$ ). The master key  $MK_j$  is defined as

$$MK_j = K^{T/w_j} \bmod M$$

In the example depicted in Fig. 1, the following master key  $MK_1$  of subset keys is given to a receiver  $u_1$ .

master key	corresponding subset keys
	$SK_{1,100}, SK_{1,110}, SK_{1,101}, SK_{1,111}$
$MK_1$	$SK_{2,100}, SK_{2,110}, SK_{2,101}$
	$SK_{5,100}, SK_{5,110}, SK_{5,101}$

**Revocation.** Transmission of secret information (e.g. a session key used for encryption or decryption of a content file) including revocation of some receivers is performed by broadcasting one or more ciphertexts. Each ciphertext is an encryption of the secret information under a subset key. To find subset keys to be used for this encryption, TC abandons all subset keys which are known to revoked receivers. It can be considered as removing all edges from the leaves corresponding to the revoked receivers to the Root in the HKT. This removal leaves one or more disjoint subtrees. Each subtree corresponds to a subset defined for its root node which is an internal node in the HKT, and each leaf of them is assigned to a non-revoked receiver. TC encrypts the secret information under the subset keys corresponding to those subsets, then broadcasts the ciphertexts. We examine the upper bound of the number of ciphertexts broadcast in section 4.1 and describe a technique used to encrypt the secret information in section 4.2.

**Decryption.** A non-revoked receiver belongs to a subset corresponding to a subtree which is left in the revocation phase, namely, the subtree contains the leaf assigned to the receiver. Note that for a non-revoked receiver  $u_j$ , there is exactly one ciphertext among the broadcast message which is an encryption under a subset key which can be derived from its master key  $MK_j$ . Naor et al. [17] introduced some techniques for listing and searching the correspondence of subsets and receivers, which can be used in conjunction with our methods.

After finding an appropriate subset, a receiver  $u_j$  computes the corresponding subset key  $SK_{k,B}$  from its master key  $MK_j$  and decrypts the ciphertext using the subset key in order to retrieve the secret information. The derivation of the subset key is performed as follows.

$$MK_j^{w_j/p_{k,B}} \bmod M = \left(K^{T/w_j}\right)^{w_j/p_{k,B}} \bmod M = K^{T/p_{k,B}} \bmod M = SK_{k,B}$$

Recall that  $p_{k,B} \mid w_j$  and  $w_j$  is a product of  $(2^{a-1} - 1) \log_a N + 1$  primes. The computational overhead is roughly  $O\left(\frac{2^a \log^5 N}{\log a}\right)$  for generation of primes as analyzed in section 4.3, and  $(2^{a-1} - 1) \log_a N$  multiplications and one modular exponentiation.

## 2.2 Method 2

### Setup

*Step 1.* The process in Step 1 is the same as in Method 1.

*Step 2.* TC chooses  $2^a - 1$  primes  $p_{b_1 b_2 \dots b_i \dots b_a}$  where  $b_i \in \{0, 1\}$  and  $\sum_{i=1}^a b_i \neq 0$ . Let  $B$  denote  $b_1 b_2 \dots b_i \dots b_a$ . TC assigns  $p_B$  to a subset  $SK_{k,B}$  defined for each internal node  $v_k$  ( $k = 1, \dots, \frac{N-1}{a-1}$ ) and publishes this assignment. Let  $T$  be a product of all primes  $p_B$ . Then TC independently chooses  $\frac{N-1}{a-1}$  elements  $K_k \in \mathbb{Z}_M^*$  and sets a subset key  $SK_{k,B}$  corresponding to a subset  $S_{k,B}$  as

$$SK_{k,B} = K_k^{T/p_B} \bmod M$$

Similar to Method 1, TC (imaginarily) gives subset keys  $SK_{k,B}$  with  $b_i = 1$  to  $i^{\text{th}}$  child node of  $v_k$ .

*Step 3.* TC gives a receiver  $u_j$  a set of  $\log_a N$  master keys  $MK_{j,k}$ , each of which is a master key of subset keys where the corresponding subsets are defined for a node  $v_k$  on  $path_j$  and those subset keys are given to its child node which is also located on  $path_j$ . A master key  $MK_{j,k}$  can derive  $2^{a-1} - 1$  subset keys. In addition, a master key  $MK_{j,1}$  can generate a subset key  $SK_{1,11\dots 1}$ . The master key  $MK_{j,k}$  is defined as

$$MK_{j,k} = K_k^{T/w_{j,k}} \bmod M$$

where  $w_{j,k}$  is a product of all primes assigned to the subsets satisfying (i) these subsets are defined for a node  $v_k$  and (ii) the corresponding subset keys are given to a child node of  $v_k$  where both  $v_k$  and the child node are located on  $path_j$  (in other words, the leaf assigned to the receiver  $u_j$  is also a leaf of a subtree rooted at the child node).

In the example depicted in Fig. 1, the following three master keys are given to a receiver  $u_1$ .

master key	corresponding subset keys
$MK_{1,1}$	$SK_{1,100}, SK_{1,110}, SK_{1,101}, SK_{1,111}$
$MK_{1,2}$	$SK_{2,100}, SK_{2,110}, SK_{2,101}$
$MK_{1,5}$	$SK_{5,100}, SK_{5,110}, SK_{5,101}$

**Revocation.** The way of transmitting secret information including revocation is the same as in Method 1.

**Decryption.** The process in this phase is basically the same as in Method 1. The only difference is the way of deriving a subset key. A receiver  $u_j$  derives a subset key  $SK_{k,B}$  from its master key  $MK_{j,k}$  as follows.

$$MK_{j,k}^{w_{j,k}/p_B} \bmod M = \left( K_k^{T/w_{j,k}} \right)^{w_{j,k}/p_B} \bmod M = K_k^{T/p_B} \bmod M = SK_{k,B}$$

Since  $w_{j,k}$  is a product of at most  $2^{a-1}$  primes, this computation requires at most  $2^{a-1} - 1$  multiplications and one modular exponentiation<sup>2</sup>. The computational overhead for generation of primes is negligible as analyzed in section 4.3.

### 3 Security of the Proposed Methods

To study the security of our methods, we investigate attacks for the Master Key technique used in these methods. The Master Key system is adopted for the whole HKT in Method 1 whereas it is applied to each internal node in the HKT in Method 2. Suppose that some attackers colluding with each other try to compromise the Master Key system in order to obtain subset keys defined in the targeted system. We consider two cases:

**Case I.** None of the attackers knows any of subset keys defined in the targeted Master Key system.

**Case II.** The attackers know at least one subset key defined in the targeted system.

A situation that all attackers are outside of  $N$  receivers is regarded as Case I in both of our methods. It is another situation regarded as Case I in Method 2 that at least one of the attackers is a receiver of this revocation scheme but no one of them is assigned to a leaf of a subtree rooted at the node where the targeted Master Key system is applied. Since  $K$  in Method 1 and  $K_k$  in Method 2 are independent of other systems, subset keys are considered to be indistinguishable from random numbers of length  $|M|$  for those attackers in Case I. Therefore we focus on Case II.

In Case II, the attackers know at least one subset key of the targeted Master Key system. Such attackers may include revoked receivers in the targeted system who attempt to obtain subset keys they do not have by breaking the system. We show that our methods are secure against any collusion of revoked receivers under the following assumption related to RSA cryptosystem.

**Assumption.** *If factors  $q_1, q_2$  of a large composite  $M = q_1 q_2$  are unknown then computing  $p^{\text{th}}$  roots (mod  $M$ ) for integral  $p > 1$  is difficult.*

We introduce a theorem and a corollary proven in the appendix to [11].

**Theorem.** *Let  $t$  and  $t_1, \dots, t_n$  be given integers and suppose there is a computable function  $F$  for which  $K^t = F(K^{t_1}, K^{t_2}, \dots, K^{t_n}) \bmod M$  for every  $K \in \mathbb{Z}_M^*$ , the group of units mod  $M$ . Let  $d = \gcd\{t_i\}$ ,  $e = \gcd(t, d)$  and  $p = d/e$ . Then we can compute  $p^{\text{th}}$  roots in  $\mathbb{Z}_M^*$ .*

<sup>2</sup> If we define a subset key  $SK_{1,11,\dots,1}$  without using the Master Key technique, we have a revocation method with  $\frac{\log N}{\log a} + 1$  keys and at most  $2^{a-2}$  multiplications at a receiver.

Suppose that such a function  $F$  exists for  $p > 1$ , then we can compute non-trivial  $p^{\text{th}}$  roots in  $\mathbb{Z}_M^*$ . This is contradictory to the assumption. Therefore the following corollary holds.

**Corollary.** *Under the assumption above, such function exists only for  $p = 1$ , namely  $\gcd\{t_i\} \mid t$ .*

Now we consider the case that some revoked receivers in the targeted Master Key system collude with each other and try to compromise a subset key which is not included in any of the master keys owned by the colluding receivers. Let  $\mathbf{G}$  be a set of master keys, which are integer power of  $K \bmod M$ ,  $\{MK_1 = K^{t_1} \bmod M, MK_2 = K^{t_2} \bmod M, \dots, MK_k = K^{t_k} \bmod M\}$ . Suppose (i) there exists a function  $F$  computing a subset key  $SK_m = K^{T/p_m} \bmod M = K^{t_m} \bmod M$  from the set  $\mathbf{G}$  and (ii)  $SK_m$  is not included in the any master key of  $\mathbf{G}$ . On one hand, from (i) and the corollary,

$$\gcd\{t_i : MK_i \in \mathbf{G}\} \mid t_m \quad (1)$$

must hold. By definition of a master key,  $t_i$  is a product of primes corresponding to the subsets which are not included in the master key  $MK_i$ . Since we can write  $t_i = \alpha_i p_m$  where  $\gcd(\alpha_i, p_m) = 1$  from (ii), we have  $\gcd\{t_i : MK_i \in \mathbf{G}\} = \alpha p_m$  where  $\gcd(\alpha, p_m) = 1$ . On the other hand, since  $T$  is a product of all primes corresponding to all subsets, we can write  $T = \beta p_m$  where  $\gcd(\beta, p_m) = 1$  (i. e.  $\beta = t_m$ ). From (I), we have  $\alpha p_m \mid \beta$ . However,  $\gcd(\alpha, p_m) = \gcd(\beta, p_m) = 1$ . This is contradictory, so the assumption that  $SK_m$  which is not included in any master key in  $\mathbf{G}$  can be derived from  $\mathbf{G}$  is wrong. This proves that our methods are secure against any conspiracy of revoked receivers under the assumption described above.

## 4 Discussions on the Proposed Methods

### 4.1 The Number of Ciphertexts

Let  $\#CT$  denote the number of ciphertexts broadcast in our methods.  $\#CT$  is equal to the number of subsets corresponding to subtrees which are left in the revocation phase. In this section we examine its upper bound.

Recall that a sender needs to broadcast one ciphertext encrypted under  $SK_{1,11,\dots,1}$  if no receivers are revoked. Now we increment the number of revoked receivers one by one. To maximize  $\#CT$  we should choose a new revoked receiver such that it shares minimum paths with receivers that have been already revoked, because the shared paths do not contribute to increase the number of the subtrees. Using this strategy, we choose up to  $a - 1$  revoked receivers as a leaf of subtrees which are rooted at distinct child nodes of the Root. Each picking of revoked receiver increases  $\#CT$  by  $\log_a N - 1$ . When we choose  $a^{\text{th}}$  revoked receiver,  $\#CT$  is increased by  $\log_a N - 2$  by choosing a leaf of a subtree rooted at the remaining child node of the Root. Similarly, each addition of  $[a^{j-1} + 1]^{\text{th}}$  to

$[a^{j-1}(a-1)]^{\text{th}}$  revoked receiver increases  $\#CT$  by  $\log_a N - j$ , and each addition of  $[a^{j-1}(a-1) + 1]^{\text{th}}$  to  $[a^j]^{\text{th}}$  revoked receiver increases it by  $\log_a N - j - 1$ . Therefore we have the upper bound of the number of ciphertexts as follows. For  $r < a$  clearly we have  $r(\log_a N - 1) + 1$ , and for  $r \geq a$  we have

$$\begin{aligned}
 & 1 + \sum_{i=1}^r (\log_a N - \lceil \log_a i \rceil) - \sum_{j=1}^{\lfloor \log_a r \rfloor} \{a^j - a^{j-1}(a-1)\} \\
 &= 1 + r \log_a N - \sum_{i=1}^r \lceil \log_a i \rceil - \sum_{j=1}^{\lfloor \log_a r \rfloor} a^{j-1} \\
 &= r \log_a N - (\lfloor \log_a r \rfloor + 1)r + a^{\lfloor \log_a r \rfloor} \\
 &< r \log_a N - r(\log_a r - 1 + 1) + a^{\log_a r} \\
 &= r(\log_a(N/r) + 1) \\
 &= r \left( \frac{\log(N/r)}{\log a} + 1 \right)
 \end{aligned}$$

Since  $r(\log_a N - 1) + 1 < r(\log_a(N/r) + 1)$  when  $1 \leq r < a$ , this upper bound holds for any  $r \geq 1$ .

## 4.2 Encryption of Secret Information

Each ciphertext broadcast in our methods is an encryption of secret information  $I$  (e.g. a session key) under a subset key. Any encryption algorithm which is considered to be secure can be used for this encryption. For example, we can use a secure block cipher algorithm with the block size  $|I|$ .

However, the length of a subset key,  $|M|$ , is equal to the length of a secure modulus of RSA cryptosystem and generally  $|M| > |BK|$ , where  $|BK|$  is the key size of the block cipher algorithm. As introduced in [17], we can use a one-way function  $h : \mathbb{Z}_M^* \rightarrow \{0, 1\}^{|BK|}$  that maps elements which are randomly distributed over  $\mathbb{Z}_M^*$  to randomly distributed strings of the desired length [19]. Namely, we can write the encryption of the secret information as  $E_{h(SK)}(I)$  where  $SK$  and  $E_{BK}(m)$  respectively denote a subset key and an encryption of message  $m$  under a block cipher algorithm using an encryption key  $BK$ . This gives that the size of each ciphertext is reduced to the size of the secret information which is transmitted, regardless of the size of a subset key.

## 4.3 Representation of Primes

In our methods a receiver needs to use some primes for derivation of a subset key. In this section we present techniques to store or find those primes and evaluate their storage and computational overhead.

**Method 1.** The total number of primes assigned to the subsets is  $(2^a - 2) \frac{N-1}{a-1} + 1$  in Method 1. Since the size of the  $n^{\text{th}}$  prime is  $O(n \log n)$  [12], we roughly



estimate that the size of each prime is at most  $O(2^a N \log 2^a N)$ . In order to derive a subset key  $SK_{k,B}$  in the decryption phase, a receiver needs to compute  $w_j/p_{k,B}$  which is a product of  $(2^{a-1} - 1) \log_a N$  primes corresponding to the subsets to which the receiver belongs except  $p_{k,B}$ . If receivers strictly store the primes which are required for derivation of subset keys, they need the storage of  $O\left(\left(\frac{2^{a-1}-1}{\log a} \log N + 1\right) (\log N + a + \log (\log N + a))\right)$  bits. Note that since those primes are public, receivers do not need to store them in a confidential manner.

On the other hand, this amount of storage overhead might be too large for some type of receivers. In order to reduce the size of such non-secret storage, we can define the assignment of the primes to the subsets as follows. A prime  $p_{k,B}$  corresponding to a subset  $S_{k,B}$  is  $(B)_2$  <sup>th</sup> smallest prime larger than  $(k-1)L$ , where  $(B)_2$  denotes a binary number represented by a bit string  $B$  and  $L$  is a positive integer. Since at most  $2^a - 1$  subsets are defined for an internal node in the HKT,  $L$  should be large so that an interval  $((k-1)L, kL]$  contains at least  $2^a - 1$  primes. If a number  $p$  is chosen at random, the probability that it is prime is about  $1/\ln p$  [23]. Recall that the size of a prime used in the method is at most  $O(2^a N \log 2^a N)$ . Therefore, if we use  $L$  satisfying  $L > (2^a - 1) \ln(2^a N \log 2^a N)$ , it is expected that the interval  $((k-1)L, kL]$  contains at least  $2^a - 1$  primes.

Each receiver can compute  $p_{k,B}$  from  $k$  and  $B$  in an on-the-fly manner as follows. From  $(k-1)L+1$ , a receiver tests each number using a primality testing algorithm until it finds  $(B)_2$  <sup>th</sup> smallest prime. An example of a probabilistic primality testing algorithm is the Miller-Rabin algorithm. Since the complexity of the algorithm for testing a number  $p$  is  $O(\log^3 p)$  [23], it is expected that the computational overhead for finding a prime is  $O(\log^4 p)$ . A receiver  $u_j$  needs to find at most  $2^a - 1$  primes (including primes  $u_j$  does not use) for each of  $\log_a N$  internal nodes on  $path_j$ , therefore the total computational overhead for generation of primes is roughly

$$O\left(\frac{2^a - 1}{\log a} \log N (\log N + a + \log (\log N + a + \log (\log N + a)))^4\right)$$

Note that we assume receivers can not store the primes strictly in order to evaluate Method 1.

**Method 2.** The total number of primes assigned to the subsets in Method 2 is  $2^a - 1$ . Note that this is much smaller than in Method 1. Since the bit length of the largest prime is roughly  $O(a + \log a)$ , the size of the storage which is required to store those primes is  $O((2^a - 1)(a + \log a))$  bits. It may be possible for receivers to store those primes strictly if a parameter  $a$  is chosen reasonably, and we assume it for evaluation of Method 2. Note that since those primes are system-wide universal and public, receivers do not have to store them in a secure non-volatile memory such as a storage for master keys, but they can store them in a usual mask ROM which is used to store program codes.

We also have some ways to reduce the size of the storage. For example, we can define the assignment of those primes such that a prime  $p_B$  corresponding to

a subset  $S_{k,B}$  is  $(B)_2^{\text{th}}$  smallest odd prime. Receivers store  $A/2$  bits table, with each bit telling them whether or not the corresponding odd number is prime, where  $A$  is large enough to consist of  $2^a - 1$  primes, such that  $A \approx 2^a a$ . This table is also system-wide universal and non-secret. This technique is introduced in [12], as well as another way to cut down the size of the storage by listing the size of gaps between primes.

As another option, receivers can compute those primes in an on-the-fly manner. It is easy to find  $2^a - 1$  smallest primes for reasonably chosen  $a$ , and it requires no storage space.

#### 4.4 Other Properties

In this section we briefly explain other properties that our methods have.

**The Number of Revoked Receivers.** It is not necessary to fix  $r$ , the number of revoked receivers, in advance in our methods. A sender can select an arbitrary group of  $r$  ( $0 \leq r < N$ ) receivers that are revoked at each time of transmission of secret information. Conversely, the sender can choose any select group of receivers as actual recipients of each transmission. The number of ciphertexts broadcast is roughly proportional to  $r$ . This is an advantage of the revocation methods using a key tree structure [5,11,16,17,24,25] including ours over the methods using a secret sharing scheme [2,15,18]. In the latter methods,  $w$  which is the upper bound of  $r$  must be fixed in advance and the length of broadcast message is  $O(w)$  regardless of the number of receivers actually revoked.

**Stateless Receivers.** In our methods, no receivers need to change their master key(s) in order to revoke or re-entitle receivers. Therefore our methods are suitable for stateless receivers. Suppose a receiver  $u_j$  has been revoked during a certain period and is re-entitled to obtain the secret information which will be transmitted afterward. Even if  $u_j$  has recorded all messages broadcast during the period and colludes with other receivers which have been also revoked during the period, it can not obtain the secret information sent at that time unless the encryption scheme used to encrypt the secret information is compromised.

**Traitor Tracing.** As described in section [1,1], a traitor tracing technology is used to find a receiver who contributed for production of a non-legitimate receiver device by giving its private keys. The requirement for tracing traitors proposed in [17] is to find the identities of those that contributed their keys to a non-legitimate receiver and revoke them with still allowing broadcasting to the legitimate receivers. Since our methods have the same property as their methods with respect to the tracing capability such as a bifurcation property, their efficient tracing algorithm is effective in conjunction with our methods. The upper bound of the bifurcation value  $z$  is  $2/3$  in our methods, therefore the algorithm can be performed with at most  $t \log_{1/z} N$  iterations, where  $t$  denotes the number of traitors.

## 5 Modification of CPPM and CPRM

CPPM and CPRM [8] are mechanisms for protection of copyrighted content recorded on pre-recorded or recordable media from unauthorized copying. Those mechanisms contain revocation of receivers (i.e. recorders or players). In this section we show a modification of them using the Master Key technique which reduces the number of keys a receiver stores.<sup>3</sup>

### 5.1 Brief Description of CPPM and CPRM

In CPPM and CPRM, Trusted Center (TC) which is a sender defines a table with  $X$  rows and  $Y$  columns and chooses a key  $K_{x,y}$  ( $x = 1, \dots, X$ ,  $y = 1, \dots, Y$ ) for each element  $(x, y)$  in the table. A compliant receiver  $u_j$  is given a unique vector  $V_j = (v_1 \dots v_Y)$  where  $v_y \in \{1, \dots, X\}$  and a set of  $Y$  keys  $K_{v_1,1}, \dots, K_{v_Y,Y}$ . On the other hand, each compliant pre-recorded or recordable medium is given a Media Key Block (MKB) on its pre-recorded area during its manufacturing process. The MKB is a collection of encryptions of a session key under a key  $K_{x,y}$ , where the session key is a key used for encryption or decryption of the content file stored on the medium. Note that the MKB does not contain the encryptions under keys  $K_{x,y}$  which are given to the revoked receivers. In consequence, the revoked receivers will not be able to obtain the session key from the medium. A non-revoked receiver stores at least one key with high probability which can be used to decrypt a ciphertext in the MKB in order to obtain the session key. This construction gives a revocation method requiring a sender to broadcast at most  $XY$  ciphertexts on the medium and each receiver to store  $Y$  keys.

### 5.2 The Modification

Now we modify this method using the Master Key technique. Instead of choosing keys  $K_{x,y}$  independently with each other, TC chooses and publishes distinct  $XY$  primes  $p_{x,y}$  for each element  $(x, y)$  in the table, and defines each key  $K_{x,y}$  as

$$K_{x,y} = K^{T/p_{x,y}} \bmod M$$

where  $M$  is a public value and a product of two large secret primes,  $K$  is a secret value chosen randomly from  $\mathbb{Z}_M^*$ , and  $T$  is a product of all primes  $p_{x,y}$ . Then TC gives a receiver  $u_j$  a master key  $MK_j$  of  $Y$  keys, one key from each column. Each of the keys corresponds to an element  $(v_y, y)$  in the table, where  $v_y$  is  $y^{\text{th}}$  element of the vector  $V_j$ . The master key  $MK_j$  is defined as

$$MK_j = K^{T/w_j} \bmod M$$

where  $w_j = \prod_{y=1}^Y p_{v_y,y}$  and  $v_y$  is  $y^{\text{th}}$  element of  $V_j$ .

<sup>3</sup> Note that discussions on the properties of CPPM and CPRM in this paper are based on section 4.5 of [17].

Given a master key  $MK_j$ , a receiver  $u_j$  can derive a key  $K_{v_y,y}$  from  $MK_j$  as

$$MK_j^{w_j/p_{v_y,y}} \bmod M = \left(K^{T/w_j}\right)^{w_j/p_{v_y,y}} \bmod M = K^{T/p_{v_y,y}} \bmod M = K_{v_y,y}$$

Discussions on the security of the proposed revocation methods in section 3 are also directly suitable for this modification. We can also use techniques described in section 4.2 and 4.3 in order to reduce the size of each ciphertext in the MKB and the size of the storage at a receiver, respectively. This modification reduces the number of keys each receiver stores to only one in exchange for additional computational overhead, i.e.  $Y - 1$  multiplications and one modular exponentiation, assuming that receivers store  $Y$  primes strictly. Other properties such that the size of the MKB and the upper bound of the number of revoked receivers still remain the same as in the original methods. The size of the MKB, namely, the number of ciphertexts in the MKB is at most  $XY$ , and the size of each ciphertext is the size of the session key. Note that as analyzed in [17], since the probability that a legitimate receiver is revoked increases non-negligibly when the number of revoked receivers becomes large, the revocation capability must be bounded by  $X$ .

## 6 Summary

In this paper we have proposed two efficient revocation methods which are suitable for stateless receivers. Our methods use the Master Key technique and the Power Set Method' with an  $a$ -ary key tree structure in order to reduce the number of keys each receiver stores and the number of ciphertexts broadcast, respectively. Method 1 requires receivers to store only one key. Method 2 is its variant which reduces the computational overhead of receivers in exchange for an increase in the number of master keys they store. We have discussed the security of our methods and some techniques used in those methods. We also have shown a modification of CPPM and CPRM using the Master Key technique where the number of keys each receiver stores is reduced to one.

## Acknowledgments

I am deeply grateful to Dan Boneh for helpful discussions and suggestions. I thank Philippe Golle for many comments regarding the paper. This work would have never started without discussions with Ryuji Ishiguro, Yoshitomo Osawa and Tateo Oishi.

## References

1. S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 239-248, 1983.

2. J. Anzai, N. Matsuzaki and T. Matsumoto, "A Quick Group Key Distribution Scheme with Entity Revocation," *Advances in Cryptology – Asiacrypt '99, Lecture Notes in Computer Science* 1716, Springer, pp. 333-347, 1999.
3. S. Berkovits, "How to Broadcast a Secret," *Advances in Cryptology – Eurocrypt '91, Lecture Notes in Computer Science* 547, Springer, pp. 535-541, 1991.
4. D. Boneh and M. Franklin, "An Efficient Public Key Traitor Tracing Scheme," *Advances in Cryptology – Crypto '99, Lecture Notes in Computer Science*, Vol. 1666, Springer-Verlag, pp. 338-353, 1999.
5. R. Canetti, T. Malkin and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption," *Advances in Cryptology – Eurocrypt '99, Lecture Notes in Computer Science* 1592, Springer, pp. 459-474, 1999.
6. G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys," *Advances in Cryptology – Crypto '89, Lecture Notes in Computer Science* 435, Springer, pp. 316-322, 1990.
7. B. Chor, A. Fiat and M. Naor, "Tracing Traitors," *Advances in Cryptology – Crypto '94, Lecture Notes in Computer Science*, Vol. 839, Springer-Verlag, pp. 257-270, 1994.
8. "Content Protection for Pre-recorded Media Specification" and "Content Protection for Recordable Media Specification," available from <http://www.4centity.com/tech/cprm/>.
9. W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, IT-22 (6), 1976.
10. A. Fiat and M. Naor, "Broadcast Encryption," *Advances in Cryptology – Crypto '93, Lecture Notes in Computer Science* 773, Springer, pp. 480-491, 1994.
11. Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups," *Proceedings of ACM Conference on Computer and Communication Security, CCS 2000*.
12. D. E. Knuth, "The Art of Computer Programming," vol. 2, Addison-Wesley, 1981.
13. R. Kumar, S. Rajagopalan and A. Sahai, "Coding Constructions for Blacklisting Problems without Computational Assumptions," *Advances in Cryptology – Crypto '99, Lecture Notes in Computer Science* 1666, Springer, pp. 609-623, 1999.
14. M. Luby and J. Staddon, "Combinatorial Bounds for Broadcast Encryptions," *Advances in Cryptology – Eurocrypt '98, Lecture Notes in Computer Science* 1403, Springer, pp. 512-526, 1998.
15. N. Matsuzaki, J. Anzai and T. Matsumoto, "Light Weight Broadcast Exclusion Using Secret Sharing," *Information Security and Privacy – 5th Australasian Conference, ACISP 2000, Lecture Notes in Computer Science* 1841, Springer, pp. 313-327, 2000.
16. D. A. McGrew and A. T. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," Manuscript, available from <http://www.csee.umbc.edu/~sherman/Papers/itse.ps>, 1998.
17. D. Naor, M. Naor and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science* 2139, Springer, pp. 41-62, 2001.
18. M. Naor and B. Pinkas, "Efficient Trace and Revoke Schemes," *Financial Cryptography '2000, Lecture Notes in Computer Science* 1962, Springer, pp. 1-20, 2000.
19. M. Naor and O. Reingold, "Number-Theoretic Constructions of Efficient Pseudo-Random Functions," *Proceedings of 38th IEEE Symposium on Foundations of Computer Science*, pp. 458-467, 1997.

20. R. Poovendran and J. S. Baras, "An Information Theoretic Analysis of Rooted-Tree Based Secure Multicast Key Distribution Schemes," *Advances in Cryptology – Crypto '99*, Lecture Notes in Computer Science 1666, Springer, pp. 624-638, 1999.
21. R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, 21, pp. 120-126, 1978.
22. A. Shamir, "How to Share a Secret," *Communications of the ACM*, 22, pp. 612-613, 1979.
23. D. R. Stinson, "Cryptography: Theory and Practice," CRC Press, 1995.
24. D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures," IETF Network Working Group, Request for Comments: 2627, available from <ftp://ftp.ietf.org/rfc/rfc2627.txt>, 1999.
25. C. K. Wong, M. Gouda and S. S. Lam, "Secure Group Communications Using Key Graphs," *Proceedings of ACM SIGCOMM '98*, 1998.

# Optimistic Mixing for Exit-Polls

Philippe Golle<sup>1</sup>, Sheng Zhong<sup>2</sup>, Dan Boneh<sup>1</sup>,  
Markus Jakobsson<sup>3</sup>, and Ari Juels<sup>3</sup>

<sup>1</sup> Stanford University, Stanford, CA 94305 USA,  
{pgolle,dabo}@cs.stanford.edu

<sup>2</sup> Yale University, New Haven, CT 06520 USA,  
sheng.zhong@yale.edu

<sup>3</sup> RSA Labs, RSA Security Inc, Bedford, MA 01730 USA,  
{mjakobsson,ajuels}@rsasecurity.com

**Abstract.** We propose a new mix network that is optimized to produce a correct output very fast when all mix servers execute the mixing protocol correctly (the usual case). Our mix network only produces an output if no server cheats. However, in the rare case when one or several mix servers cheat, we convert the inputs to a format that allows “back-up” mixing. This back-up mixing can be implemented using any one of a wide array of already proposed (but slower) mix networks. When all goes well, our mix net is the fastest, both in real terms and asymptotically, of all those that offer standard guarantees of privacy and correctness. In practice, this benefit far outweighs the drawback of a comparatively complex procedure to recover from cheating. Our new mix is ideally suited to compute almost instantly the output of electronic elections, whence the name “exit-poll” mixing.

## 1 Introduction

The recently devised mix network constructions of Furukawa and Sako [FS01] and Neff [Nef01] provide the full spectrum of security properties desirable in an election scheme. They achieve privacy, which is to say concealment of individual votes, and also robustness against Byzantine server failures. They additionally possess the property of *universal verifiability*, that is, the ability for any entity to verify the correct functioning of the mix, even in the face of an adversary that controls all servers and voters. Finally, the Furukawa/Sako and Neff mixes are substantially more efficient in terms of both computational and communications requirements than previously proposed mix networks with similar security properties.

Fast as they are, however, these mixes still remain cumbersome as tools for large-scale elections. Sako *et al.* report a running time of roughly six hours to process a batch of 100,000 votes [EMMOS02]. In a federal election involving large precincts (conceivably millions of ballots in some states) a complete tally would thus require many hours. Premature media predictions of Gore’s victory in Florida in the 2000 U.S. presidential election demonstrate the hunger of the electorate for timely information, and also the mischief that can be wrought

in its absence. There is clearly a political and social need for faster tallying mechanisms than Furukawa/Sako and Neff alone can provide.

We describe here a mix network that is tailored for election systems, but with a substantial speedup over Furukawa/Sako and Neff. In settings like that described by Sako *et al.*, for example, we estimate that our construction is capable of yielding a six-to-eight times speedup. We achieve this improvement by taking an “optimistic” or “fast-track” [GRR98] approach. In particular, we identify functionality in Furukawa/Sako and Neff that is not needed in the likely case that mix servers behave correctly and that most ballots are well formed. In the optimistic case, we show how to dispense with the costly property of robustness against Byzantine server failures. We also provide a form of universal verifiability that is somewhat weaker than the standard definition, but less costly, and adequate for nearly all types of elections, as we explain.

We refer to our proposal as an *exit-poll* mix network, by analogy with the “exit polls” used to provide fast predictions of election outcomes. If servers behave correctly, our exit-poll mix yields a correct election tally very rapidly. We expect this to be by far the most common case. If server cheating occurs, our mix identifies misbehaving servers. The privacy of all votes remains protected (given a majority of honest servers), but our mix does not produce an output. In such cases, our exit-poll scheme permits seamless fallback to a more heavyweight mix (like Furukawa/Sako or Neff) which can take over, complete the mixing and produce an output. Such heavyweight mixes can also be employed to achieve supplemental, after-the-fact certification of an election tally achieved with our mix.

Our exit-poll mix is a general ciphertext-to-plaintext scheme. While it is designed particularly for use in election schemes, we note that it can be employed in many of the other applications for which such mix networks are useful. Examples include anonymous e-mail [Cha80] and bulletin boards, anonymous payment systems [JM98] as well as anonymized Web browsing [GGMM97].

The rest of the paper is organized as follows. We review related work in section 2. In section 3, we describe ElGamal re-encryption mix networks. We present the high-level design of our new mix network in section 4, and give a detailed description of the protocol in section 5. In section 6, we prove the properties of our mix net. We conclude in section 7.

## 2 Related Work

Chaum proposed the first mix network, a decryption mix, in [Cha80]. In Chaum’s construction, users encrypt their inputs with the public-key of each mix server, starting with the last and ending with the first mix server in the net. Each mix server removes one layer of encryption, until the plaintexts are output by the last server. The weakness of this approach is that the mixing can not proceed if a single server is unavailable. To achieve robustness against server failures, [PIK93] introduced a new type of mix, re-encryption mix nets, in which the mixing and decryption phases are separated (see section 3). The particular re-



Scheme	Re-encrypt	<b>Proof and verification</b>	Decrypt	Addition chain speed-up?
Cut and choose <sup>2</sup> [SK95, OKST97]	$2n$	$642nk$	$(2 + 4k)n$	no
Pairwise permutation [Abe99, JJ99]	$2n$	$7n \log n(2k - 1)$	$(2 + 4k)n$	partially
Matrix representation [FS01]	$2n$	$18n(2k - 1)$	$(2 + 4k)n$	partially
Polynomial scheme [Nef01]	$2n$	$8n(2k - 1)$	$(2 + 4k)n$	partially
Exit-poll mixing [this paper]	$6n$	$6 + 12k$	$(5 + 10k)n$	yes

**Fig. 1.** Optimistic cost per server (for a total of  $k$  servers) of mixing  $n$  items with different mix schemes, measured in number of exponentiations. Note that our proof and verification costs do not depend on  $n$ . The table also indicates whether addition chains can be used to pre-compute exponentiations. “Partially” indicates that addition chains can be used only in the mixing phase but not to prove correctness.

encryption mix of [PIK93] was shown insecure in [PP89, Pfi94], but was later fixed by [OKST97].

The main difficulty of re-encryption mixes is to design computationally efficient ways for mix servers to *prove* that they mixed and re-encrypted their inputs correctly in the mixing phase. The first techniques were based on costly general purpose cut-and-choose zero-knowledge proofs [SK95, OKST97, Abe98]. Milimix [JJ99] and MIP-2 [Abe99, AH01] are based on more efficient zero-knowledge proofs specifically designed to prove that an output is a re-encryption of one of two inputs.

The most efficient schemes to date that offer the full spectrum of security properties are those of Furukawa and Sako [FS01] and Neff [Nef01]. The table in figure 1 compares the real cost of mixing  $n$  items (in terms of the number of exponentiations required) with different mixing schemes (the numbers are taken from the respective papers). The column indicating the cost of proof and verification is in bold, because that is typically by far the most expensive step, and it is the step that we are optimizing. The cost of re-encryption is higher in our scheme than in others, but the difference pales in comparison with our savings in the proof and verification step. Furthermore, the re-encryption exponentiations can be pre-computed. The table also indicates whether each mixing scheme can take advantage of the speed-up techniques proposed in [Jak99] for multiple exponentiations with respect to a fixed base. These techniques, based on addition chains, reduce the equivalent cost of one exponentiation to approximately 10 multiplications for reasonable sizes of batches (see [Jak99] for more details). This amounts to a very significant speed-up. Our scheme is not only the fastest, but also the only one that can fully take advantage of addition chains in this sense.

<sup>1</sup> Other aspects of that proposal were later found flawed, and corrected, in [MK00].

The exposed vulnerabilities do not affect the soundness of the speed-up techniques.

<sup>2</sup> We note that these proposals have computational costs quadratic in the number of servers, due to the use of interactive proofs. However, if non-interactive proofs are employed – as in subsequent papers – this is brought down to a linear cost. The computational cost we use in the table assumes that this enhancement is performed.

An attractive alternative to mix networks is homomorphic encryption, in particular the Paillier scheme [Pai99]. Election schemes based on homomorphic encryption require a good deal of computation for verification of correct ballot formation, but very little for tallying. In practice, therefore, they can be much faster than mix-based election schemes. Until recently, an objection to homomorphic schemes has been their inability to accommodate write-in votes, an unavoidable requirement in the election systems of many jurisdictions. Kiayias and Yung have recently devised a simple scheme that circumvents this difficulty [KY02]. In brief, the idea is to permit each ballot to contain either a standard vote or a write-in vote, and to set aside write-in votes for separate processing via a mix network (in the unlikely case that this is needed).

It is our belief that mix networks will nonetheless remain an essential tool in electronic voting, as they still provide features that homomorphic schemes cannot. Vote-buying and coercion are serious threats in any election, but potentially much more problematic in Internet-based elections, given the anonymizing mechanisms available on the Internet and its reach across many jurisdictions. Schemes based on mix networks offer ways of minimizing these threats [HS00], while homomorphic schemes do not. A second advantage of mix networks is their flexibility with regard to key distribution. To distribute shares in the Paillier system without use of a trusted third party requires expensive joint RSA key-generation protocols (e.g., [BF97]), and distribution of a fresh RSA modulus for every election involving a different distribution of trust. Mix-based schemes can be based on discrete-log cryptosystems, with simpler and more generalizable keying mechanisms. With this in mind, we propose a new mix network which offers a significant efficiency improvement over existing constructions.

### 3 ElGamal Re-encryption Mix Network

In this section, we describe the basic operation of a plain-vanilla re-encryption mix network based on the ElGamal cryptosystem. It will serve as a basis for our main construction described in section 4 and 5. The operation of a mix network can be divided into the following steps:

1. **Setup Phase.** In the setup phase, the mix servers jointly generate the public and private parameters of an ElGamal cryptosystem. The private key is shared in a  $(t, n)$ -threshold verifiable secret sharing scheme among all mix servers, while the public parameters are published.
2. **Submission of Inputs.** All users submit their inputs to the mix encrypted with the public parameters generated in the setup phase.
3. **Mixing Phase.** Each mix server in turn mixes and re-randomizes the batch of ciphertexts submitted to the mix.
4. **Decryption Phase.** After the mixing is done, all output ciphertexts are decrypted by a quorum of mix servers.

We start with a description of the ElGamal cryptosystem, and discuss in particular how to re-randomize ciphertexts in the mixing phase. We then explain

how to jointly generate the parameters for an ElGamal cryptosystem in the setup phase, and how the quorum decryption works in the decryption phase.

### 3.1 ElGamal Cryptosystem

ElGamal is a randomized public-key encryption scheme. Let  $P$  and  $Q$  be two large primes such that  $P = 2Q + 1$ . We denote by  $G_Q$  the subgroup of  $\mathbb{Z}_P^*$  of order  $Q$ . Let  $g$  be a generator of  $G_Q$ . The private key is an element  $x \in \mathbb{Z}_Q$ , and the corresponding public key is  $y = g^x \bmod P$ . To encrypt a plaintext  $m \in G_Q$ , we choose a random element  $r \in \mathbb{Z}_Q$  and compute the ciphertext  $E_y(m, r) = (g^r, my^r)$ . Note that an ElGamal ciphertext is a pair of elements of  $G_Q$ . To get the decryption  $D_x(G, M)$  of an ElGamal ciphertext  $(G, M)$ , we compute  $D_x(G, M) \triangleq M/G^x$ . The ElGamal cryptosystem is semantically secure [Bon98] if the decisional Diffie-Hellman assumption holds in the group  $G_Q$ .

### Re-randomization

ElGamal is a randomized encryption scheme that allows for re-randomization of ciphertexts. Given an ElGamal ciphertext  $(G, M)$ , a mix server can efficiently compute a new ciphertext  $(G', M')$  that decrypts to the same plaintext as  $(G, M)$  (we say that the ciphertext  $(G', M')$  is a re-randomization of  $(G, M)$ ). To re-randomize a ciphertext, the mix server chooses a value  $r \in \mathbb{Z}_Q$  uniformly at random and computes  $(G', M') = (Gg^r, My^r)$ . Observe that this does not require knowledge of the private key, and that the exponentiation can be pre-processed.

Given two ElGamal ciphertexts, it is infeasible to determine whether one is a re-randomization of the other without knowledge of either the private decryption key  $x$  or the re-randomization factor  $r$ , assuming that the Decision Diffie-Hellman problem is hard in  $G_Q$ . A mix server can use this property to hide the correspondence between its input and output ciphertexts: the input ciphertexts are first re-randomized, then output in a random order.

However, a mix server who knows the re-randomization factor  $r$  can efficiently convince a verifier that  $(G', M')$  is a re-randomization of  $(G, M)$  without revealing  $r$ . The proof of re-randomization consists simply of proving that  $\log_g(G'/G) \equiv \log_y(M'/M) \pmod{P}$ , which trivially implies that there exists  $r$  such that  $(G', M') = (Gg^r, My^r)$ . To prove the former discrete logarithm equality, we may use for example Schnorr signatures [Sch91] (as suggested in [Jak98]) or a non-interactive version [FS86] of the Chaum-Pedersen protocol [CP92]. This proof of re-randomization will serve as the basis for a proof that allows a mix server to prove that it mixed its inputs correctly (observe that in the real proof of correctness, a mix server must not reveal which output is a re-randomization of which input, so the proof outlined above will not work *as is*).

### 3.2 Distributed ElGamal

In the setup phase, the mix servers jointly generate the parameters  $(P, Q, g, x, y)$  of an ElGamal cryptosystem, in such a way that the private key  $x$  is distributed in a  $(n, t)$ -threshold verifiable secret sharing (VSS) scheme among all

mix servers [Fel87]. To set up this VSS, a simple solution is to have a trusted “dealer” generate all the parameters and then distribute shares of the private key to the mix servers. (An alternative solution that does not require a trusted third party was proposed in [Ped91], but it was later found flawed by [GJK+99]. Note that the proved-correct protocol suggested in [GJK+99] is for a different VSS scheme.)

With the private key thus shared, it is known that any quorum of  $t$  mix servers can jointly decrypt the output ElGamal ciphertexts without explicitly reconstructing the private key  $x$ . A quorum  $T$  of  $t$  servers can decrypt a ciphertext  $(G, M)$  as follows:

$$D_x(G, M) = \frac{M}{G^x} = \frac{M}{\prod_{j \in T} (G^{x_j})^{\prod_{l \in T, l \neq j} \frac{-1}{j-l}}}.$$

Observe that this equation requires each server  $j \in T$  to raise  $G$  to the  $x_j$ -th power. Server  $j$  may prove that it has honestly computed  $S = G^{x_j}$  with the following proof of discrete logarithm equality:  $\log_G S \equiv \log_g y_j (= x_j) \pmod{P}$ .

## 4 Mix Net Design

Our new mix net mixes ciphertexts like an ElGamal re-encryption mix. The novelty lies first in a highly efficient method for proving that the mixing was done correctly, and second in a method for falling back on a more heavyweight mix if cheating by a server is detected. We start with a high-level description of these two building blocks.

Each input ciphertext submitted to our mix net is required to be the encryption of a plaintext that includes a cryptographic checksum. To verify that a mix server operated correctly, we ask for a proof that the product of the plaintexts corresponding to the input ciphertexts equals the product of the plaintexts corresponding to the output ciphertexts. As we shall show, such proofs can be produced and verified highly efficiently without knowledge of the plaintexts. We call this proof a *proof-of-product (POP) with checksum*.

This proof however does not detect all types of cheating. Rather, it guarantees that if the mix server did not mix correctly, it had to introduce in the output at least one new ciphertext that corresponds to a plaintext with an invalid checksum. When outputs are decrypted, invalid checksums are traced to one of two sources: either an input that was originally submitted to the mix network with an invalid checksum, or a cheating mix server. The difficulty of this approach lies in the fact that since invalid checksums can only be traced at decryption time, cheating may not be detected until after the harm is done. In effect, a cheating server may be able to match inputs to outputs before cheating gets detected in the verification step. If we were to use this mix just like that, nothing could be done after a server has cheated to restore the privacy of those users whose inputs have already been traced. In particular, a second round of mixing wouldn't help.

To address this difficulty, we introduce the second main contribution of this paper, which may be of interest on its own. Our approach is to encrypt users' inputs twice (a technique we call *double enveloping*). In the verification step outlined above, the output ciphertexts are first decrypted only once. If the verification succeeds and no servers are found to have cheated, the output ciphertexts are decrypted one more time and yield the plaintext. If on the other hand one or several servers are found to have cheated, the output ciphertexts are not decrypted further. Instead, they become the input to a different (slower) mix network such as [Nef01] and are mixed a second time before being finally decrypted. This second round of mixing ensures that the privacy of users can not be compromised. A cheating server in the first round of mixing may learn at most the relationship between a double-encrypted ciphertext and a single-encrypted ciphertext, which does not help to find the corresponding plaintext after the second round of mixing.

In the rest of this section, we describe these two building blocks in greater detail.

#### 4.1 Proof of Product with Checksum

Consider a mix server who receives as inputs  $n$  ElGamal ciphertexts  $(G_i, M_i)$ , and outputs a permuted re-randomization of these, namely a permutation of the list of  $(G'_i, M'_i) = (G_i g^{r_i}, M_i y^{r_i})$ . Our key idea is to let the mix server prove that its operations are *product preserving*, i.e. that the product of the plaintexts corresponding to the input ciphertexts  $(G_i, M_i)$  equals the product of the plaintexts corresponding to the output ciphertexts  $(G'_i, M'_i)$ . The following property of the ElGamal encryption scheme makes this possible:

**Proposition 1.** (*Multiplicative Homomorphism of ElGamal*): Let  $(G_1, M_1)$  and  $(G_2, M_2)$  be ElGamal encryptions of plaintexts  $P_1$  and  $P_2$ . Then  $(G_1 G_2, M_1 M_2)$  is an ElGamal encryption of the product  $P_1 P_2$ . We call  $(G_1 G_2, M_1 M_2)$  the “product” of  $(G_1, M_1)$  and  $(G_2, M_2)$ .

Proposition 1 shows that any verifier can compute an ElGamal encryption  $(G, M)$  of  $\prod m_i$ , and an ElGamal encryption  $(G', M')$  of  $\prod m'_i$ , where  $m_i$  (resp.  $m'_i$ ) is the plaintext corresponding to  $(G_i, M_i)$  (resp.,  $(G'_i, M'_i)$ ). To prove that its operations are *product preserving*, the mix server need only prove that  $\log_g(G'/G) = \log_y(M'/M)$ . As we saw in section 3.1 this implies that  $\prod m_i = \prod m'_i$ .

#### The Need for a Checksum

The product equality  $\prod m_i = \prod m'_i$  clearly does not imply that the sets  $\{m_i\}_{i=1}^n$  and  $\{m'_i\}_{i=1}^n$  are equal. In other words, the property of being product-preserving does not by itself guarantee that a mix net operates correctly. Our approach is to restrict the plaintexts  $m_i$  (and therefore also  $m'_i$ ) to a particular format, in such a way that it becomes infeasible for a dishonest mix server to find a set  $\{m'_i\} \neq \{m_i\}$  such that  $\prod m_i = \prod m'_i$  and all the elements  $m'_i$  are of the required format. We propose to define this special format by adding a cryptographic

checksum to the plaintext, drawing on the techniques of [JJ01]. This is done as follows.

Users format their inputs to the mix net as an ElGamal encryption of a plaintext  $m$  and an ElGamal encryption of  $h(m)$ , where  $h : \{0, 1\}^* \rightarrow G_Q$  is a cryptographic hash function (in the proof of security, we model  $h$  as a random oracle [BR93]):

$$\left( E_y(m, r), E_y(h(m), r') \right)$$

Each input to the mix now consists of a *pair* of ElGamal ciphertexts. The mix re-randomizes separately each of the two ElGamal ciphertexts in every pair, then outputs all the pairs in a random order. The mix must then prove that the products of the plaintexts corresponding to the first element in the pair are the same in the input and the output ( $\prod m_i = \prod m'_i$ ) and also that the products of the plaintexts corresponding to the second element in the pair are the same in the input and the output ( $\prod h(m_i) = \prod h(m'_i)$ ). As we shall prove in section 6, these two proofs together guarantee the set equality  $\{m_i\} = \{m'_i\}$ .

## 4.2 Double Enveloping

As we have already pointed out, a mix whose correctness was enforced only by a proof-of-product with redundancy may not detect server cheating until after the harm is done. To illustrate how users' privacy may be compromised even if all cheating servers are disqualified, we offer the following example. Assume that the first mix server is corrupt and that the input submitted by user  $i$  is  $(E_y(m_i, r_i), E_y(h(m_i), r'_i))$ . The corrupt first server can replace the input of user 1 by  $(E_y(m_1, r_1)E_y(m_2, r_2), E_y(h(m_1), r'_1)E_y(h(m_2), r'_2))$  (recall the definition of the product of ElGamal ciphertexts in Section 4.1), and replace the input of user 2 by  $(1, 1, 1)$ . Such cheating will only be detected after the decryption phase. Even if the cheating server were to be disqualified and the mixing protocol restarted, the cheating server would still be able to distinguish the plaintexts submitted by users 1 and 2 from other users' plaintexts, by comparing the output of the restarted protocol with that of the first execution.

To defend against this attack, we add a second layer of encryption to the plaintext  $m$  of a user. A user whose plaintext input is  $m$  is required to submit the following triple of ciphertexts to the mix:

$$(E_y(G, r), E_y(M, r'), E_y(h(G, M), r'')),$$

where  $(G, M) = (g^{\hat{r}}, m y^{\hat{r}})$ , and as before  $h : \{0, 1\}^* \rightarrow G_Q$  is a cryptographic hash function.

Thus  $(G, M)$  replaces  $m$  in the description of POP-with-checksum above. (Other double enveloping designs resulting in the same functional structure are possible. We choose this one for concreteness.) If cheating is caused by a corrupt server, we can re-randomize all the inner-layer encryptions and their order with a standard ElGamal-based re-randomization mix net, before they are finally decrypted to plaintexts. Although the adversary might be able to link some

inner-layer encryptions to the input ciphertexts, he cannot link the final output plaintexts to them.

## 5 Exit-Poll Mix Net

**Assumptions.** We assume that there exists a bulletin board, which is accessible to the public, and is authenticated, tamper-proof, and resistant to denial-of-service attacks. All messages and proofs are posted on this bulletin board.

**Setup.** The mix servers jointly generate parameters  $(P, Q, g, x, y)$  for an ElGamal cryptosystem  $E$ . The public parameters are made public, while the private key  $x$  is shared among the mix servers in a  $(t, n)$ -threshold VSS scheme. Users are required to submit their input  $m_i$  to the mix net formatted as follows:

1. The user encrypts the input  $m_i$  to produce  $E_y(m_i) = (G_i, M_i)$ .
2. The user computes  $H_i = h(E_y(m_i))$ . As explained earlier, we model  $h$  as a random oracle in the proof of security. In practice, a publicly available hash function such as MD5 [Riv92] or SHA-1 [N95] should be used.
3. The user submits the triple  $E_y(G_i), E_y(M_i), E_y(H_i)$ . The mix servers check that every component belongs to  $G_Q$ , and that this input has not already been submitted. If any component is not in  $G_Q$ , the user is disqualified and the triple is discarded. If the same input has already been submitted by another user, the duplicate submission is discarded.
4. The user proves his knowledge<sup>3</sup> of  $G_i, M_i, H_i$ . This is important to prevent a user from re-encrypting and re-posting another user's input. This proof of knowledge should be bound to a unique mix-session identifier to achieve security across multiple invocations of the mix. Any user who fails to give the proof is disqualified, and the corresponding input is discarded.
5. We note that dishonest users may submit inputs that are not properly formatted, in the sense that the equality  $H_i = h(E_y(m_i))$  does not hold. We stress that such improperly formatted inputs can *not* force our mix net to default to the slower back-up mixing. The only event that can trigger a default to the back-up mixing is cheating by one of the mix servers.

**First Stage: Re-randomization and Mixing.** This step proceeds as in all re-randomization mix nets based on ElGamal. One by one, the mix servers re-randomize all the inputs and their order. (Note that the components of triples are not separated from each other during the re-randomization.) In addition, each mix net must give a proof that the product of the plaintexts of all its inputs equals the product of the plaintexts of all its outputs.

---

<sup>3</sup> We note that for reasons of efficiency, it suffices that he proves knowledge of one of these components, and make the proof relative to the other two. This can be done (as is standard) by letting the latter two be part of the input to the random oracle that sets the challenge for the proof.

1. Each mix server reads from the bulletin board the list of triples corresponding to re-encryptions of  $E_y(G_i), E_y(M_i), E_y(H_i)$  output by the previous mix server:  $\{(g^{r_i}, a_i \cdot y^{r_i}), (g^{s_i}, b_i \cdot y^{s_i}), (g^{t_i}, c_i \cdot y^{t_i})\}_{i=1}^N$ . (Note that even if some servers have cheated, the ciphertexts can still be formatted like that, provided that every component belongs to  $G_Q$ .)
2. The mix server re-randomizes the order of these triples according to a secret and random permutation. Note that it is the order of triples that is re-randomized, and that the three components  $E_y(G_i)$ ,  $E_y(M_i)$  and  $E_y(H_i)$  that make up each triple remain in order.
3. The mix server then re-randomizes each component of each triple independently, and outputs the results:  $\{(g^{r'_i}, a'_i \cdot y^{r'_i}), (g^{s'_i}, b'_i \cdot y^{s'_i}), (g^{t'_i}, c'_i \cdot y^{t'_i})\}_{i=1}^N$ .
4. The mix server proves that  $\prod a_i = \prod a'_i$  and  $\prod b_i = \prod b'_i$  and  $\prod c_i = \prod c'_i$ .

## Second Stage: Decryption of the Inputs

1. A quorum of mix servers jointly decrypt each triple of ciphertexts to produce the values  $G_i$ ,  $M_i$  and  $H_i$ , using the technique we reviewed in Section 3.2.
2. All triples for which  $H_i = h(G_i, M_i)$  are called *valid*.
3. Invalid triples are investigated according to the procedure described below. If the investigation proves that all invalid triples are *benign* (only users cheated), we proceed to step 4. Otherwise, the decryption is aborted and we continue with the back-up mixing.
4. A quorum of mix servers jointly decrypts the ciphertexts  $(G_i, M_i)$  for all valid triples. This successfully concludes the mixing. The final output is defined as the set of plaintexts corresponding to valid triples.

**Special Step: Investigation of Invalid Triples.** The investigation proceeds as follows. The mix servers must reveal the path of each invalid triple through the various permutations. For each invalid triple, starting from the last server, each server reveals which of its inputs corresponds to this triple, and how it re-randomized this triple. The cost of checking the path of an invalid triple is three exponentiations per mix server (the same cost as that incurred to run one input through the mix net). One of two things may happen:

- **Benign Case (Only Users Cheated):** if the mix servers successfully produce all such paths, the invalid triples are known to have been submitted by users. The decryption is resumed after the incorrect elements have been removed.
- **Serious Case (One or More Servers Cheated):** if one or more mix servers fail to recreate the paths of invalid triples, these mix servers are accused of cheating and replaced, and our mix terminates without producing an output. In this case, the inputs are handed over to the back-up mixing procedure described next.

Note that when the mix servers investigate an invalid triple we assume implicitly that the successive permutations applied by mix servers define a unique path for each triple through the mix net. This is not strictly true if two or more triples encode the same inner-layer ciphertext. Indeed if two triples correspond to



different outer-layer encryptions of the same inner-layer ciphertext, the (outer-layer) re-encryption of one can be passed off as a re-encryption of the other. In this case, the permutations do not strictly commit mix servers to a unique path for each triple. Observe however that this does not affect the investigation of invalid triples. A corrupt server may substitute one copy (i.e. outer-layer encoding) of an invalid triple for another, but must eventually account for all copies.

**Back-Up Mixing.** The *outer-layer* encryption of the inputs posted to the mix net is decrypted by a quorum of mix servers. The resulting set of *inner-layer* ciphertexts becomes the input to a standard re-encryption mix net based on ElGamal (using, for example, Neff’s scheme described in [Nef01]). At the end of this second mixing, the ciphertexts are finally decrypted to plaintexts, which concludes the mixing.

## 6 Security Analysis

We start with a brief discussion of the efficiency of our scheme. The costs are as follows for a batch consisting of  $n$  inputs:

- Re-encryption and Mixing: **Linear** number of modular exponentiations ( $6n$ ).
- Proof of Correct Mixing: **Constant** number of modular exponentiations (but number of modular multiplications linear in  $n$ ).
- Verification: **Constant** number of modular exponentiations per server (but number of modular multiplications linear in  $n$ ). The cost is also linear in the number of servers.
- Decryption: **Linear** number of modular exponentiations ( $((5 + 10k)n$  for  $k$  servers).

This makes our mix not only twice as fast as the next fastest mix network [Nef01], but also the only mix (among mixes with standard security guarantees) for which the costs are incurred mostly in the re-encryption and decryption phases. This is important because these two phases (unlike the proof phase) can benefit from the significant speed-up techniques developed in [Jak99]. Using addition chains, we estimate that the cost of one exponentiation is roughly equivalent to 10 multiplications, with reasonably sized batches.

We now turn to proving that our mix network offers guarantees of correctness, verifiability and privacy.

**Proposition 2.** (*Correctness*) *If all parties follow the protocol, the output of the mix net is a permuted decryption of the input.*

Since the set of plaintexts is preserved in re-randomizations, this follows from the correctness of decryption.

The verifiability of our mix net is a restricted form of universal verifiability in the sense that only the operation of the mix net on *valid* inputs (i.e., the inputs that are well-formed according to our protocol) are universally verifiable. We call this restricted form of verifiability “public verifiability”.

**Definition 1.** (*Public Verifiability*) A mix net is publicly verifiable if there exists a polynomially bounded verifier that takes as input the transcript of the mixing posted on the bulletin board, outputs “valid” if the set of valid outputs is a permuted decryption of all valid inputs, and otherwise outputs “invalid” with overwhelming probability. Note that to prove public verifiability, we consider an adversary that can control all mix servers and all users.

**Proposition 3.** Our mix net is publicly verifiable if there exists a group  $G$  in which the discrete logarithm problem is hard.

*Proof.* The proof proceeds by contradiction. We assume that one or several mix servers cheat during the execution of a mixing protocol, yet manage to produce a transcript that fools an outside verifier into believing that the mixing was done correctly. We show how to use these cheating mix servers to compute discrete logarithms in the group  $G$ . Our proof is based on the following lemma:

**Lemma 1.** Let  $a$  and  $b$  be two elements of a group  $G$  of order  $|G|$ . For random values  $r_1, \dots, r_N$  and  $s_1, \dots, s_N$ , we compute the following group elements:  $h_i = a^{r_i} b^{s_i}$ . Consider an adversary who on inputs  $h_1, \dots, h_N$  outputs integers  $e_1, \dots, e_N$  such that  $\prod_{i=1}^N h_i^{e_i} = 1$  and at least one of the  $e_i$ ’s is non-zero. With probability  $1 - 1/|G|$ , the knowledge of these  $e_i$ ’s allows us to compute  $\log_a b$ .

*Proof.* If  $\sum_{i=1}^N s_i e_i \neq 0$ , then we can compute  $\log_a b = -(\sum_{i=1}^N r_i e_i) / (\sum_{i=1}^N s_i e_i)$ . It remains to prove that  $\sum_{i=1}^N s_i e_i \neq 0$  happens with probability  $1 - 1/|G|$ . Since the values  $r_i$ ’s are random, the knowledge of the  $h_i$ ’s yields no information to the adversary about the  $s_i$ ’s. Indeed we have  $\log h_i = r_i + \log_a b s_i$ . Since the distribution of  $r_i$  is uniformly random, the distribution of  $s_i$  is also uniformly random given  $h_i$ . The probability that the vector  $E = (e_1, \dots, e_N)$  chosen by the adversary is orthogonal to an unknown random  $S = (s_1, \dots, s_N)$  is  $1 - 1/|G|$ .  $\square$

Now let us turn to the proof of proposition 3. We denote the inputs to the mix network as

$$(E_y(G_1, r_1), E_y(M_1, r'_1), E_y(H_1, r''_1)), \dots, (E_y(G_N, r_N), E_y(M_N, r'_N), E_y(H_N, r''_N))),$$

and denote the outputs of the mix network as

$$(E_y(\overline{G_1}, \overline{r_1}), E_y(\overline{M_1}, \overline{r'_1}), E_y(\overline{H_1}, \overline{r''_1})), \dots, (E_y(\overline{G_N}, \overline{r_N}), E_y(\overline{M_N}, \overline{r'_N}), E_y(\overline{H_N}, \overline{r''_N}))).$$

For cheating to escape detection, the equation

$$\prod_i H_i = \prod_i \overline{H_i} \tag{1}$$

must hold, and in addition we must have  $\overline{H_i} = h(\overline{G_i}, \overline{M_i})$  for all  $i$ . Furthermore, since we restrict the notion of universal verifiability to valid inputs, we have  $H_i = h(G_i, M_i)$  for all  $i$ . Equation [1](#) can therefore be rewritten:

$$\prod_i h(G_i, M_i) = \prod_i h(\overline{G_i}, \overline{M_i}). \tag{2}$$

Now recall that in our security proof, we model the hash function  $h$  as a random oracle. Each time a mix server queries  $h$  on a new input, we choose random values  $r_i$  and  $s_i$  and return  $a^{r_i}b^{s_i}$  (we answer queries on inputs that have already been queried consistently). Since the mix server cheated, equation 2 gives us a non-trivial product relationship of the type that allows us to compute discrete logarithms in the group  $G$  according to lemma 1, and this concludes the proof.  $\square$

Our mix network offers the same guarantee of privacy as all mix networks based on ElGamal re-encryptions, e.g. [Nef01].

## 7 Conclusions

We constructed a verifiable mix network that is extremely fast in case none of the mix servers cheat. This enables election officials to quickly announce the results in the common case when all mix servers honestly follow the mixing protocol. In case one or more of the mix servers cheat, our system detects the cheating server or servers and then redoes the mixing using one of the standard (slower) mix systems [Nef01]. We emphasize that server cheating cannot compromise user privacy; it just causes the mixing process to run slower.

Our fast verifiable mixing is achieved by using the homomorphic property of ElGamal encryption to quickly test that the product of all plaintext inputs is equal to the product of all plaintext outputs. Clearly, this simple product test is insufficient for proving correct mixing. However, we are able to prove that by adding an appropriate checksum to all inputs this product test becomes sufficient. Furthermore, we use double enveloping to ensure user privacy in case one or more mix servers cheat. We hope that our approach can be used to speed-up other secure distributed computations in case all participants honestly follow the protocol, without affecting security in case of cheating.

## References

- Abe98. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *Proc. of Eurocrypt '98*, pp. 437-447. Springer-Verlag, 1998. LNCS 1403.
- Abe99. M. Abe. Mix-networks on permutation networks. In *Proc. of Asiacrypt '99*, pp. 258-273, 1999. LNCS 1716.
- AH01. M. Abe and F. Hoshino. Remarks on mix-networks based on permutation networks. In *Proc. of PKC'01*.
- Bon98. D. Boneh. The Decision Diffie-Hellman Problem. In *ANTS-III*, pp. 48-63, 1998. LNCS 1423.
- BF97. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proc. of CRYPTO'97*, pp. 425-439. LNCS 1294.
- BR93. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of CCS'93*, pp. 62-73.
- Cha80. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, 24(2):84-88, 1981.

- CP92. D. Chaum and T. Pedersen. Wallet databases with observers. In *Proc. of Crypto'92*, pp. 89-105. Springer-Verlag, 1993. LNCS 740.
- DF89. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proc. of Crypto'89*, pp. 307-315. LNCS 435.
- DK00. Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. In *Proc. of Eurocrypt'2000*, pp. 557-572. LNCS 1807.
- Fel87. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. of the 28th IEEE Symposium on the Foundations of Computer Science*, IEEE Press, 1987, pp. 427-437.
- FMMOS02. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Pre-proceedings of Financial Crypto'02*.
- FS86. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proc. of CRYPTO'86*, 1987.
- FS01. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of Crypto '01*, pp. 368-387. Springer-Verlag, 2001. LNCS 2139.
- GGMM97. E. Gabber, P. Gibbons, Y. Matias and A. Mayer. How to make personalized Web browsing simple, secure and anonymous. In *Proc. of Financial Cryptography'97*, pp. 17-31, 1997.
- GJK+99. R. Gennaro, S. Jarecki and H. Krawczyk and T. Rabin, Secure Distributed Key Generation for Discrete-Log Based Cryptosystems, In *Proc. of Eurocrypt '99*, pp. 295-310, 1999. LNCS 1592.
- GRR98. R. Gennaro, M. Rabin and T. Rabin. Simplified VSS and fast-track multi-party computations with applications to threshold cryptography. In *Proc. ACM PODC'98*, pp. 101-111.
- HS00. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. of Eurocrypt'00*, pp. 539-556. Springer-Verlag, 2000. LNCS 1807.
- Jak98. M. Jakobsson. A practical mix. In *Proc. of Eurocrypt '98*, pp. 448-461. Springer-Verlag, 1998. LNCS 1403.
- JM98. M. Jakobsson and D. M'Raihi. Mix-based electronic payments. In *Proc. of SAC'98*, pp. 157-173. Springer-Verlag, 1998. LNCS 1556.
- Jak99. M. Jakobsson. Flash mixing. In *Proc. of PODC '99*, pp. 83-89. ACM, 1999.
- JJ99. M. Jakobsson and A. Juels. Millimix: mixing in small batches. DIMACS Technical Report 99-33.
- JJ01. M. Jakobsson and A. Juels. An optimally robust hybrid mix network. In *Proc. of PODC'01*, pp. 284-292. ACM Press. 2001.
- JJR02. M. Jakobsson, A. Juels and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. To be presented at USENIX'02.
- KY02. A. Kiayias and M. Yung. Self-tallying elections and perfect ballot secrecy. In *Proc. of PKC'02*, pp. 141-158. LNCS 2274.
- MK00. M. Mitomo and K. Kurosawa. Attack for flash mix. In *Proc. of Asiacrypt'00*, pp. 192-204. LNCS 1976.
- Nef01. A. Neff. A verifiable secret shuffle and its application to E-Voting. In *Proc. of ACM CCS'01*, pp. 116-125. ACM Press, 2001.
- N95. NIST. Secure hash standard. Federal Information Processing Standards Publication 180-1. 1995.
- OKST97. W. Ogata, K. Kurosawa, K. Sako and K. Takatani. Fault tolerant anonymous channel. In *Proc. of ICICS '97*, pp. 440-444, 1997. LNCS 1334.

- Pai99. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of Eurocrypt'99*, pp. 223-238. LNCS 1592.
- Ped91. T. Pedersen. A Threshold cryptosystem without a trusted party. In *Proc. of Eurocrypt'91*, pp. 522-526, 1991.
- PIK93. C. Park, K. Itoh and K. Kurosawa. Efficient anonymous channel and all/nothing election Scheme. In *Proc. of Eurocrypt '93*, pp. 248-259. Springer-Verlag, 1993. LNCS 765.
- PP89. B. Pfitzmann and A. Pfitzmann. How to break the direct RSA-implementation of mixes. In *Proc. of Eurocrypt '89*, pp. 373-381. Springer-Verlag, 1989. LNCS 434.
- Pfi94. B. Pfizmann. Breaking an efficient anonymous channel. In *Proc. of Eurocrypt'94*, pp. 339-348.
- Riv92. R. Rivest. The MD5 message-digest algorithm. IETF Network Working Group, RFC 1321, 1992.
- SK95. K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *Proc. of Eurocrypt '95*. Springer-Verlag, 1995. LNCS 921.
- Sch91. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161-174, 1991.
- TY98. Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In *Proc. of PKC'98*.

# Improved Construction of Nonlinear Resilient S-Boxes

Kishan Chand Gupta and Palash Sarkar

Cryptology Research Group, Applied Statistics Unit, Indian Statistical Institute,  
203, B.T. Road, Kolkata 700108, India,  
kishan.t@isical.ac.in, palash@isical.ac.in

**Abstract.** We provide two new construction methods for nonlinear resilient functions. The first method is a simple modification of an elegant construction due to Zhang and Zheng and constructs  $n$ -input,  $m$ -output resilient S-boxes with degree  $d > m$ . We prove by an application of the Griesmer bound for linear error correcting codes that the modified Zhang-Zheng construction is superior to the previous method of Cheon in Crypto 2001. Our second construction uses a sharpened version of the Maiorana-McFarland technique to construct nonlinear resilient functions. The nonlinearity obtained by our second construction is better than previously known construction methods.

**Keywords:** S-box, Griesmer bound, Resiliency, nonlinearity, algebraic degree, stream cipher.

## 1 Introduction

An  $(n, m)$  S-box (or vectorial function) is a map  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ . By an  $(n, m, t)$  S-box (or  $(n, m, t)$ -resilient function) we mean  $t$ -resilient  $(n, m)$  S-box. An  $(n, 1, t)$ -resilient S-box is a resilient Boolean function. The cryptographic properties (like resiliency, nonlinearity, algebraic degree) of Boolean functions necessary for stream cipher applications have already been extensively studied. The resiliency property of S-box was introduced by Chor et al [5] and Bennett et al [1]. However, to be used in stream ciphers several other properties of S-box like nonlinearity and algebraic degree are also very important. Stinson and Massey [18] considered nonlinear resilient functions but only to disprove a conjecture.

It was Zhang and Zheng [20] who first proposed a beautiful method of transforming a linear resilient S-box to construct a nonlinear resilient S-box with high nonlinearity and high algebraic degree keeping cryptography in mind. After that, serious efforts to construct nonlinear S-box with high nonlinearity and high algebraic degree has been made [8, 7, 12, 4](see Section 2.4).

The current state-of-art in resilient S-box design can be classified into the following two approaches.

1. Construction of  $(n, m, t)$ -resilient functions with very high nonlinearity.

## 2. Construction of $(n, m, t)$ -resilient functions with degree $d > m$ and high nonlinearity.

The first problem has been studied in [20, 8, 7, 12]. The currently best known results are obtained using the construction described in [12], though in certain cases, for small number of variables, the search technique of [7] yields better results. The second problem has been less studied. To the best of our knowledge, the only known construction which provides functions of the second type is due to Cheon [4].

In this paper, we first prove that the correlation immunity of a resilient function is preserved under composition with an arbitrary Boolean function. This property is useful for possible application of resilient S-boxes in designing secure stream ciphers. Our main contribution consists of two different constructions for the above two classes of problems. In both cases our results provide significant improvement over all previous methods.

The construction for the second problem is a simple modification of the Zhang-Zheng method [20]. To get algebraic degree  $d > m$ , we start with an  $[n, d+1, t+1]$  code. Then we apply Zhang-Zheng construction to obtain a nonlinear S-box. Finally we drop  $d+1-m$  output columns to obtain an  $(n, m, t)$ -resilient S-box (see Section 4). This simple modification is powerful enough to improve upon the best known construction with algebraic degree greater than  $m$  [4]. This clearly indicates the power of the original Zhang-Zheng construction. Our contribution is to apply the Griesmer bound for linear error correcting codes to *prove* that the modified Zhang-Zheng construction is superior to the best known construction [4]. We know of no other work where such a provable comparison of construction has been presented.

The Maiorana-McFarland technique is a well known method to construct nonlinear resilient functions. The idea is to use affine functions on small number of variables to construct nonlinear resilient functions on larger number of variables. We provide a construction to generate functions of the first type using a sharpened version of the Maiorana-McFarland method. For Boolean functions, the Maiorana-McFarland technique to construct resilient functions was introduced by Camion et al [2]. Nonlinearity calculation for the construction was first performed by Seberry, Zhang and Zheng [16]. This technique was later sharpened by Chee et al [3] and Sarkar-Maitra [15]. For S-boxes this technique has been used by [7] and [12], though [7] uses essentially a heuristic search technique. Here we develop and sharpen the technique of affine function concatenation to construct nonlinear resilient S-boxes. This leads to significant improvement in nonlinearity over that obtained in [12]. Thus we obtain better results than [12] which currently provides the best known nonlinearity results for most choices of input parameters  $n, m, t$ .

The paper is organized as follows. Section 2 provides basic definitions, notations, theory needed and a quick review of recent construction. In Section 3 we prove the composition theorem. Section 4 provides modified Zhang-Zheng construction and some theorems to prove its advantage over Cheon construction. Section 5 provide some definitions and theory needed in that section. It

also provides a construction by which we get  $(n, m, t)$ -resilient S-box with non-linearity greater than the nonlinearity obtained in [12] which is known to be best till date. In Section 6 we compare modified Zhang-Zhang construction with Cheon construction, and also compare Construction-I of Section 5 with Pasalic and Maitra construction [12]. Section 7 concludes this paper.

## 2 Preliminaries

This section has four parts. We cover preliminaries on Boolean functions and S-boxes in Sections 2.1 and 2.2 respectively. In Section 2.3, we mention the coding theory result that we require. In Section 2.4, we summarize the previous construction results.

### 2.1 Boolean Functions

Let  $F_2 = GF(2)$ . We consider the domain of a Boolean function to be the vector space  $(F_2^n, \oplus)$  over  $F_2$ , where  $\oplus$  is used to denote the addition operator over both  $F_2$  and the vector space  $F_2^n$ . The inner product of two vectors  $u, v \in F_2^n$  will be denoted by  $\langle u, v \rangle$ . The weight of an  $n$ -bit vector  $u$  is the number of ones in  $u$  and will be denoted by  $wt(u)$ . The (Hamming) distance between two vectors  $x = (x_1, x_2, \dots, x_n)$  and  $y = (y_1, y_2, \dots, y_n)$  is the number of places where they differ and is denoted by  $d(x, y)$ . The Walsh Transform of an  $m$ -variable Boolean function  $g$  is an integer valued function  $W_g : \{0, 1\}^m \rightarrow [-2^m, 2^m]$  defined by (see [9, page 414])

$$W_g(u) = \sum_{w \in F_2^m} (-1)^{g(w) \oplus \langle u, w \rangle}. \quad (1)$$

The Walsh Transform is called the spectrum of  $g$ . The inverse Walsh Transform is given by

$$(-1)^{g(u)} = \frac{1}{2^m} \sum_{w \in F_2^m} W_g(w) (-1)^{\langle u, w \rangle}. \quad (2)$$

An  $m$ -variable function is called *correlation immune* of order  $t$  ( $t$ -CI) if  $W_g(u) = 0$  for all  $u$  with  $1 \leq wt(u) \leq t$  [17, 19]. Further the function is balanced if and only if  $W_g(0) = 0$ . A balanced  $t$ -CI function is called  *$t$ -resilient*. For even  $n$ , an  $n$ -variable function  $f$  is called *bent* if  $W_f(u) = \pm 2^{\frac{n}{2}}$ , for all  $u \in F_2^n$  (see [14]). This class of functions is important in both cryptography and coding theory.

A parameter of fundamental importance in cryptography is the non-linearity of a function (see [9]). This is defined to be the distance from the set of all affine functions. It is more convenient to define it in terms of the spectrum of a Boolean function. The non-linearity  $nl(f)$  of an  $n$ -variable Boolean function  $f$ , is defined as

$$nl(f) = 2^{n-1} - \frac{1}{2} \max_{u \in F_2^n} |W_f(u)|.$$

For even  $n$ , bent functions achieve the maximum possible nonlinearity.



A Boolean function  $g$  can be uniquely represented by a multivariate polynomial over  $F_2$ . The degree of the polynomial is called the algebraic degree or simply the degree of  $g$ .

## 2.2 S-Boxes

An  $(n, m)$  S-box (or vectorial function) is a map  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ . Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be an S-box and  $g : \{0, 1\}^m \rightarrow \{0, 1\}$  be an  $m$ -variable Boolean function. The composition of  $g$  and  $f$ , denoted by  $g \circ f$  is an  $n$ -variable Boolean function defined by  $(g \circ f)(x) = g(f(x))$ . An  $(n, m)$  S-box  $f$  is said to be  $t$ -CI, if  $g \circ f$  is  $t$ -CI for every non-constant  $m$ -variable linear function  $g$  (see [20]). Further, if  $f$  is balanced then  $f$  is called  $t$ -resilient. ( The function  $f$  is said to be balanced if  $g \circ f$  is balanced for every non-constant  $m$ -variable linear function  $g$  ). By an  $(n, m, t)$  S-box we mean  $t$ -resilient  $(n, m)$  S-box. Let  $f$  be an  $(n, m)$  S-box. The nonlinearity of  $f$ , denoted by  $nl(f)$ , is defined to be

$$nl(f) = \min\{nl(g \circ f) : g \text{ is a non-constant } m\text{-variable linear function}\}.$$

Similarly the algebraic degree of  $f$ , denoted by  $deg(f)$ , is defined to be

$$deg(f) = \min\{deg(g \circ f) : g \text{ is a non-constant } m\text{-variable linear function}\}.$$

We will be interested in  $(n, m)$  S-boxes with maximum possible nonlinearity. If  $n = m$ , the S-boxes achieving the maximum possible nonlinearity are called maximally nonlinear [6]. If  $n$  is odd, then maximally nonlinear S-boxes have nonlinearity  $2^{n-1} - 2^{\frac{n-1}{2}}$ . For even  $n$ , it is possible to construct  $(n, m)$  S-boxes with nonlinearity  $2^{n-1} - 2^{\frac{n}{2}}$ , though it is an open question whether this value is the maximum possible.

An  $(n, m)$  S-box with nonlinearity  $2^{n-1} - 2^{\frac{n}{2}-1}$  is called perfect nonlinear S-box. Nyberg [10] has shown that perfect nonlinear functions exist if and only if  $n$  is even and  $n \geq 2m$ . For odd  $n \geq 2m$ , it is possible to construct S-boxes with nonlinearity  $2^{n-1} - 2^{\frac{n-1}{2}}$ .

If we fix an enumeration of the set  $\{0, 1\}^n$ , then an  $(n, m)$  S-box  $f$  is uniquely defined by a  $2^n \times m$  matrix  $M_f$ . Given a sequence of S-boxes  $f_1, \dots, f_k$ ; where  $f_i$  is an  $(n_i, m)$  S-box we define the *concatenation* of  $f_1, \dots, f_k$  to be the matrix

$$M = \begin{bmatrix} M_{f_1} \\ M_{f_2} \\ \vdots \\ M_{f_k} \end{bmatrix}.$$

If  $2^{n_1} + \dots + 2^{n_k} = 2^n$  for some  $n$ , then the matrix  $M$  uniquely defines an  $(n, m)$  S-box  $f$ . In this case we say  $f$  is the *concatenation* of  $f_1, \dots, f_k$ .

## 2.3 Coding Theory Results

We will use some standard coding theory results and terminology all of which can be found in [9]. An  $[n, k, d]$  binary linear code is a subset of  $F_2^n$  which is a vector space of dimension  $k$  over  $F_2$  having minimum distance  $d$ . We here

mention the Griesmer bound (see [9, page 546]). For an  $[n, k, d]$  linear code let  $N(k, d) =$  length of the shortest binary linear code of dimension  $k$  and minimum distance  $d$ .

The Griesmer bound states (see [9, page 547])

$$N(k, d) \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{2^i} \right\rceil. \quad (3)$$

We say that the parameters  $n, k, d$  satisfy the Griesmer bound with equality if  $n = \sum_{i=0}^{k-1} \left\lceil \frac{d}{2^i} \right\rceil$ . There is a general construction (see [9, page 550]) which gives large class of codes meeting the Griesmer bound with equality. Given  $d$  and  $k$ , define  $s = \left\lceil \frac{d}{2^{k-1}} \right\rceil$  and  $d = s2^{k-1} - \sum_{i=1}^p 2^{u_i-1}$  where  $k > u_1 > \dots > u_p \geq 1$ . Given  $d$  and  $k$ , there is an  $[n = s(2^k - 1) - \sum_{i=1}^p (2^{u_i} - 1), k, d]$  code meeting the Griesmer bound with equality if  $\sum_{i=1}^{\min(s+1, p)} u_i \leq sk$  (see [9, page 552]). This condition is satisfied for most values of  $d$  and  $k$ .

## 2.4 Some Recent Constructions

Here we summarize the previous construction results.

1. Zhang and Zheng [20]: This is the first paper to provide an elegant general construction of nonlinear resilient S-boxes. The main result proved is the following [20, Corollary 6]. If there exists a linear  $(n, m, t)$ -resilient function, then there exists a nonlinear  $(n, m, t)$ -resilient function with algebraic degree  $(m - 1)$  and nonlinearity  $\geq (2^{n-1} - 2^{n-\frac{m}{2}})$ .
2. Kurosawa, Satoh and Yamamoto [8, Theorem 18]: For any even  $l$  such that  $l \geq 2m$ , if there exists an  $(n - l, m, t)$ -resilient function, then there exists an  $(n, m, t)$ -resilient function, whose nonlinearity is at least  $2^{n-1} - 2^{n-\frac{l}{2}-1}$ .
3. Johansson and Pasalic [7]: They use a linear error correcting code to build a matrix  $A$  of small affine functions. Resiliency and nonlinearity is ensured by using non-intersecting codes along with the matrix  $A$ . The actual non-intersecting codes used were obtained by a heuristic search technique. It becomes difficult to carry out this search technique for  $n > 12$ .
4. Pasalic and Maitra [12]: They use the matrix  $A$  of the previous method (3) along with highly nonlinear functions for their construction. The nonlinearity obtained is higher than the previous methods, except in certain cases, where the search technique of (3) yields better results.
5. Cheon [4, Theorem 5]: Uses linearized polynomial to construct nonlinear resilient function. The nonlinearity calculation is based on Hasse-Weil bound for higher genus curves. The main result is the following. If there exists  $[n, m, t]$  linear code then for any non-negative integer  $D$  there exists a  $(n+D+1, m, t-1)$ -resilient function with algebraic degree  $D$  and nonlinearity at least  $(2^{n+D} - 2^n \lfloor \sqrt{2^{n+D+1}} \rfloor + 2^{n-1})$ . *To date, this is the only construction which provides  $(n, m, t)$  nonlinear resilient S-boxes with degree greater than  $m$ .*

### 3 A Composition Theorem for $S$ -boxes

We consider the composition of an  $(n, m)$   $S$ -box and an  $m$ -variable Boolean function. The following result describes the Walsh Transform of the composition.

**Theorem 1.** *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  and  $g : \{0, 1\}^m \rightarrow \{0, 1\}$ . Then for any  $w \in F_2^n$ ,*

$$W_{(g \circ f)}(w) = \frac{1}{2^m} \sum_{v \in F_2^m} W_g(v) W_{(l_v \circ f)}(w)$$

where  $l_v = \langle v, x \rangle$  and  $(l_v \circ f)(x) = \langle v, f(x) \rangle$ .

*Proof.* By Equation 2, we have  $(-1)^{g(x)} = \frac{1}{2^m} \sum_{w \in F_2^m} W_g(w) (-1)^{\langle w, x \rangle}$ .

Hence,

$$\begin{aligned} (-1)^{(g \circ f)(x)} &= (-1)^{g(f(x))} = \frac{1}{2^m} \sum_{v \in F_2^m} W_g(v) (-1)^{\langle v, f(x) \rangle} \\ &= \frac{1}{2^m} \sum_{v \in F_2^m} W_g(v) (-1)^{(l_v \circ f)(x)}. \end{aligned}$$

By Equation 1, we have

$$\begin{aligned} W_{g \circ f}(w) &= \sum_{x \in F_2^n} (-1)^{(g \circ f)(x) \oplus \langle w, x \rangle} = \frac{1}{2^m} \sum_{x \in F_2^n} \sum_{v \in F_2^m} W_g(v) (-1)^{(l_v \circ f)(x) \oplus \langle w, x \rangle} \\ &= \frac{1}{2^m} \sum_{v \in F_2^m} W_g(v) \sum_{x \in F_2^n} (-1)^{(l_v \circ f)(x) \oplus \langle w, x \rangle} = \frac{1}{2^m} \sum_{v \in F_2^m} W_g(v) W_{(l_v \circ f)}(w) \quad \square \end{aligned}$$

**Corollary 1.** *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be a balanced  $S$ -box. Let  $g$  be an  $m$ -variable Boolean function. Then  $(g \circ f)$  is balanced if and only if  $g$  is balanced.*

*Proof.* Since  $f$  is balanced,  $W_{(l_v \circ f)}(w) = 0$  for all nonzero  $v \in F_2^m$ .

Thus  $W_{g \circ f}(0) = \frac{1}{2^m} W_g(0) 2^m = W_g(0)$ .  $\square$

**Remark:** It is possible for  $(g \circ f)$  to be balanced even when either only  $f$  is unbalanced or both  $f$  and  $g$  are unbalanced. We present examples for these cases. Let  $f : \{0, 1\}^3 \rightarrow \{0, 1\}^2$  be an unbalanced  $S$ -box and  $f_1, f_2$  are component functions.

(a) Let  $f_1(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_2x_3$  and  $f_2(x_1, x_2, x_3) = x_2 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2x_3$  and  $g(x_1, x_2) = x_1 \oplus x_2$ . Here  $f$  is unbalanced but  $g$  is balanced. Observe  $(g \circ f)(x_1, x_2, x_3) = f_1(x_1, x_2, x_3) \oplus f_2(x_1, x_2, x_3) = x_1 \oplus x_2x_3$  is balanced.

(b) Let  $f_1(x_1, x_2, x_3) = x_3 \oplus x_1x_2 \oplus x_1x_2x_3$  and  $f_2(x_1, x_2, x_3) = x_2 \oplus x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_2x_3$  and  $g(x_1, x_2) = x_1x_2$ . Here both  $f$  and  $g$  are unbalanced. Observe  $(g \circ f)(x_1, x_2, x_3) = f_1(x_1, x_2, x_3)f_2(x_1, x_2, x_3) = x_3$ , which is balanced.

Theorem 1 and Corollary 1 provide the following theorem.

**Theorem 2.** *Let  $f$  be a  $t$ -resilient  $S$ -box and  $g$  be any arbitrary Boolean function then  $(g \circ f)$  is  $t$ -CI. Further  $(g \circ f)$  is  $t$ -resilient if and only if  $g$  is balanced.*

Theorem 2 shows that correlation immunity of an  $(n, m, t)$ -resilient  $S$ -box is preserved under composition with an *arbitrary*  $m$ -variable Boolean function. This is an important security property for the use of resilient  $S$ -boxes in stream cipher design.

## 4 Construction of $(n, m, t)$ -Resilient $S$ -Box with Degree $> m$ .

In this section we modify an elegant construction by Zhang and Zheng [20] to obtain high degree nonlinear resilient  $S$ -boxes. The following result is well known (see for example [20]).

**Theorem 3.** *Let  $C$  be a  $[n, m, t + 1]$  binary linear code. Then we can construct an linear  $(n, m, t)$ -resilient function.*

### Modified Zhang-Zheng (MZZ) Construction.

- Inputs: Number of input columns =  $n$ , number of output columns =  $m$ , degree =  $d \geq m$  and resiliency =  $t$ .
- Output: An  $(n, m, t)$ -resilient function having degree  $d$  and nonlinearity  $2^{n-1} - 2^{n-\lceil \frac{d+1}{2} \rceil}$ .

### Procedure

1. Use an  $[n, d + 1, t + 1]$  code to obtain an  $(n, d + 1, t)$ -resilient function  $f$ .
2. Define  $g = G \circ f$ , where  $G : \{0, 1\}^{d+1} \rightarrow \{0, 1\}^{d+1}$  is a bijection and  $\deg(G) = d$ ,  $nl(G) \geq 2^d - 2^{\lfloor \frac{d+1}{2} \rfloor}$  [11]. Then  $nl(g) \geq 2^{n-d-1}(2^d - 2^{\lfloor \frac{d+1}{2} \rfloor}) = 2^{n-1} - 2^{n-\lceil \frac{d+1}{2} \rceil}$  and  $\deg(g) = d$  [20, Corollary 6].
3. Drop  $(d + 1 - m)$  columns from the output of  $g$  to obtain an  $(n, m, t)$ -resilient function with degree  $d$  and nonlinearity  $2^{n-1} - 2^{n-\lceil \frac{d+1}{2} \rceil}$ .

**Remark:** For Step 2 above, there are other bijections by which we get the same value of  $nl(G)$  but  $\deg(G) = d$  is achieved only for  $G$  obtained from the inverse mapping  $\tau : GF(2^{d+1}) \rightarrow GF(2^{d+1})$ , with  $\tau(x) = x^{-1}$  [6].

The modification to the Zhang-Zheng construction is really simple. If we want degree  $d$ , then we start with an  $[n, d + 1, t + 1]$  code. Then we apply the main step of Zhang-Zheng construction to obtain a nonlinear  $S$ -box. Finally we drop  $d + 1 - m$  output columns to obtain an  $(n, m, t)$ -resilient  $S$ -box. Though simple, this modification is powerful enough to improve upon the best known construction with high algebraic degree [4]. This shows the power of the original Zhang-Zheng construction. Our contribution is to *prove* by an application of the Griesmer bound that the MZZ construction is superior to the best known construction [4, Cheon]. We know of no other work where such provable comparisons of construction has been presented.

**Theorem 4.** *Let  $n, m, d, t$  be such that the following two conditions hold.*

1. *Either (a)  $d < m$  or (b)  $d \geq m \geq \log_2(t+1)$ .*
  2. *The parameters  $n, d+1, t+1$  meet the Griesmer bound with equality.*
- Then it is not possible to construct an  $(n, m, t)$ -resilient function  $f$  with degree  $d$  using Cheon [4] method.*

*Proof.* Recall the Cheon construction from Section 2.4. Given any  $[N, M, T+1]$  and a non negative integer  $D$ , the Cheon construction produces an  $(N+D+1, M, T)$ -resilient function with degree  $D$ . Thus if  $f$  is obtained by the Cheon construction we must have  $n = N+D+1$ ,  $m = M$ ,  $t = T$  and  $d = D$ .

This means that an  $[n-d-1, m, t+1]$  code will be required by the Cheon construction. Since the parameters  $n, d+1, t+1$  satisfies Griesmar bound with equality we have  $n = \sum_{i=0}^d \lceil \frac{t+1}{2^i} \rceil$ .

*Claim:* If (a)  $d < m$  or (b)  $d \geq m \geq \log_2(t+1)$  then  $n-d-1 < \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$ .

*Proof of the Claim:* Since  $n = \sum_{i=0}^d \lceil \frac{t+1}{2^i} \rceil$  we have that  $n-d-1 < \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$  if and only if  $\sum_{i=0}^d \lceil \frac{t+1}{2^i} \rceil - d - 1 < \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$ . If  $d < m$ , then the last mentioned condition is trivially true. So suppose  $d \geq m \geq \log_2(t+1)$ . Then the above inequality holds if and only if  $\sum_{i=m}^d \lceil \frac{t+1}{2^i} \rceil < d+1$ . Since  $m \geq \log_2(t+1)$ ,  $\sum_{i=m}^d \lceil \frac{t+1}{2^i} \rceil = d-m+1 < d+1$  for  $m \geq 1$ . This completes the proof of the claim.

Since  $n-d-1 < \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$ , the parameters  $n-d-1, m, t+1$  violate the Griesmer bound and hence an  $[n-d-1, m, t+1]$  code do not exist. Thus Cheon method cannot be used to construct the function  $f$ .  $\square$

The following result is a consequence of Theorem 4 and the MZZ construction.

**Theorem 5.** *Let  $n, m, d, t$  be such that the following two conditions hold.*

1. *Either (a)  $d < m$  or (b)  $d \geq m \geq \log_2(t+1)$ .*
  2. *An  $[n, d+1, t+1]$  code meeting the Griesmer bound with equality exist.*
- Then it is possible to construct an  $(n, m, t)$ -resilient function  $f$  with degree  $d$  by the MZZ method which cannot be constructed using Cheon [4] method.*

**Remark:** As mentioned in [9, page 550] there is a large class of codes which meet the Griesmer bound with equality. Further, the condition  $d \geq m \geq \log_2(t+1)$  is quite weak. Hence there exists a large class of  $(n, m, t)$ -resilient functions which can be constructed using MZZ construction but cannot be constructed using Cheon [4] construction. See Section 6 for some concrete examples.

*Nonlinearity in Cheon method is  $(2^{N+D} - 2^N \lfloor \sqrt{2^{N+D+1}} \rfloor + 2^{n-1})$  (see item 5 of Section 2.4) which is positive if  $D \geq N+1$  for  $N \geq 2$ . So for  $D \leq N$ , Cheon method do not provide any nonlinearity. Thus Cheon method may provide high algebraic degree but it does not provide good nonlinearity. In fact, in the next theorem we prove that nonlinearity obtained by MZZ method is larger than nonlinearity obtained by Cheon method.*

**Theorem 6.** *Let  $f$  be an  $(n, m, t)$ -resilient function  $f$  of degree  $d$  and nonlinearity  $n_1$  constructed by Cheon method. Suppose there exists a linear  $[n, d+1, t+1]$  code. Then it is possible to construct an  $(n, m, t)$ -resilient function  $g$  with degree  $d$  and nonlinearity  $n_2$  using MZZ method. Further  $n_2 \geq n_1$ .*

*Proof.* Since  $[n, d+1, t+1]$  code exists, the MZZ construction can be applied to obtain an  $(n, m, t)$ -resilient function  $g$  with degree  $d$  and nonlinearity  $nl(g) = n_2 = 2^{n-1} - 2^{n-\lceil \frac{d+1}{2} \rceil}$ . It remains to show that  $n_2 \geq n_1$ , which we show now. Recall  $n_1 = 2^{n-1} - 2^{n-d-1} \lfloor \sqrt{2^n} \rfloor + 2^{n-d-2}$ . Hence  $n_2 - n_1 \geq -2^{n-\frac{d+1}{2}} + 2^{n-d-1} \lfloor \sqrt{2^n} \rfloor - 2^{n-d-2}$ . Thus we have  $n_2 \geq n_1$  if  $-2^{\frac{-(d+1)}{2}} + 2^{-(d+1)} \lfloor \sqrt{2^n} \rfloor - 2^{-(d+2)} \geq 0$ . The last condition holds if and only if  $\lfloor \sqrt{2^n} \rfloor \geq 2^{d+1} (\frac{1}{2^{\frac{d+1}{2}}} + \frac{1}{2^{d+2}})$ .

So  $n_2 \geq n_1$  if  $\sqrt{2^n} - 1 \geq 2^{\frac{d+1}{2}} + 2^{-1}$ . i.e. if  $2^{\frac{n}{2}} \geq 2^{\frac{d+1}{2}} + \frac{3}{2}$ . Again the last condition hold for  $1 \leq d \leq n-3$ . Hence  $n_2 \geq n_1$  for  $1 \leq d \leq n-3$ . The maximum possible degree of an S-box is  $n-1$ . For  $d = n-1$  and  $d = n-2$ , Cheon construction requires  $[0, m, t+1]$  and  $[1, m, t+1]$  codes respectively. Clearly such code do not exist. Hence  $n_2 \geq n_1$  holds for all  $d$ .  $\square$

**Lemma 1.** *Let  $f$  be an  $(n, m, t)$ -resilient function  $f$  of degree  $d \geq m$  constructed by Cheon method and  $m \geq \log_2(t+1)$ . Then the parameters  $n, d+1, t+1$  satisfy the Griesmer bound.*

*Proof.* Since  $f$  has been obtained from Cheon method, there exists an  $[n-d-1, m, t+1]$  code. Hence the parameters  $n-d-1, m$  and  $t+1$  satisfy the Griesmar bound. Since  $n-d-1, m$  and  $t+1$  satisfy the Griesmar bound we have  $n-d-1 \geq \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$ . i.e. we have  $n \geq d+1 + \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$ . As  $m \geq \log_2(t+1)$  we have  $\lceil \frac{t+1}{2^i} \rceil = 1$  for  $i \geq m$ . Hence  $n \geq (d+1) - (d-m+1) + \sum_{i=m}^d \lceil \frac{t+1}{2^i} \rceil + \sum_{i=0}^{m-1} \lceil \frac{t+1}{2^i} \rceil$ . This shows  $n \geq m + \sum_{i=0}^d \lceil \frac{t+1}{2^i} \rceil$  and consequently  $n \geq \sum_{i=0}^d \lceil \frac{t+1}{2^i} \rceil$ . Thus the parameters  $n, d+1, t+1$  satisfy the Griesmer bound.  $\square$

**Remark:** Since the parameters  $n, d+1$  and  $t+1$  satisfy the Griesmer bound, in most cases it is possible to obtain an  $[n, d+1, t+1]$  code (see [9, page 550]) and apply Theorem 6. In fact we do not know any case where a function can be constructed using the Cheon method but not by the MZZ method. Theorems 5 and 6 prove the clear advantage of the MZZ method over the Cheon construction. Thus MZZ method is the currently known best method to construct  $[n, m, t]$ -resilient function with degree  $d > m$ .

## 5 A Construction to Obtain High Nonlinearity

In this section we concentrate on obtaining  $(n, m, t)$ -resilient S-boxes with high nonlinearity only. We present a construction method which improves the nonlinearity obtainable by the previously known methods. We start by mentioning the following result which is restatement of Lemma 7 in [7].

**Theorem 7.** *Let  $C$  be a  $[u, m, t+1]$  code. Then it is possible to construct  $(2^m - 1) \times m$  matrix  $D$  with entries from  $C$ , such that,  $\{c_1 D_{i,1} \oplus \dots \oplus c_m D_{i,m} : 1 \leq i \leq 2^m - 1\} = C \setminus \{(0, \dots, 0)\}$  for each nonzero vector  $(c_1, \dots, c_m) \in F_2^m$ .*

Let  $D$  be the matrix in Theorem 7. For  $(1 \leq i \leq 2^m - 1)$  and  $(1 \leq j \leq m)$  define a  $u$ -variable linear function  $L_{i,j}(x_1, \dots, x_u) \triangleq \langle D_{i,j}, (x_1, \dots, x_u) \rangle$ . Given the code  $C$  we define a  $(2^m - 1) \times m$  matrix  $L(C)$  whose entries are  $u$ -variable linear functions by defining the  $i, j$  th entry of  $L(C)$  to be  $L_{i,j}(x_1, \dots, x_u)$ . We have the following result which follows directly from Theorem 7.

**Proposition 1.** *Let  $c \in F_2^m$  be a nonzero row vector. Then all the entries of the column vector  $L(C)c^T$  are distinct.*

For positive integers  $k, l$  with  $k \leq l$ , we define  $L(C, k, l)$  to be the submatrix of  $L(C)$  consisting of the rows  $k$  to  $l$ . Thus  $L(C, 1, 2^m - 1) = L(C)$ . Let  $G(y_1, \dots, y_p)$  be a  $(p, m)$  S-box whose component functions are  $G_1, \dots, G_m$ . We define  $G \oplus L(C, k, l)$  to be an  $(l - k + 1) \times m$  matrix whose  $i, j$  th entry is  $G_j(y_1, \dots, y_p) \oplus L_{k+i-1,j}(x_1, \dots, x_u)$  for  $1 \leq i \leq l - k + 1$  and  $1 \leq j \leq m$ . If  $l - k + 1 = 2^r$  for some  $r$  then  $G \oplus L(C, k, l)$  defines an S-box  $F : \{0, 1\}^{r+p+u} \rightarrow \{0, 1\}^m$  in the following manner.

$$F_j(z_1, \dots, z_r, y_1, \dots, y_p, x_1, \dots, x_u) = G_j(y_1, \dots, y_p) \oplus L_{k+i-1,j}(x_1, \dots, x_u)$$

where  $1 \leq j \leq m$ ,  $1 \leq i \leq 2^r$ ,  $F_1, \dots, F_m$  are the component functions of  $F$  and  $z_1 \dots z_r$  is the binary representation of  $i - 1$ . By  $F = G \oplus L(C, k, l)$  we will mean the above representation of the S-box  $F$ . Note that the function  $F$  is  $t$ -resilient, since each  $L_{i,j}(x_1, \dots, x_u)$  is non-degenerate on at least  $(t + 1)$  variables and hence  $t$ -resilient.

In the matrix  $M = G(y_1, \dots, y_p) \oplus L(C, k, l)$  we say that the row  $L_{i,*}$  of  $L(C)$  is repeated  $2^p$  times. Let  $G(y_1, \dots, y_p)$  and  $H(y_1, \dots, y_q)$  be  $(p, m)$  and  $(q, m)$  S-boxes respectively and  $M_1 = G \oplus L(C, k, l)$ ,  $M_2 = H \oplus L(C, k, l)$ . Then we say that the row  $L_{i,*}$  of  $L(C)$ , ( $k \leq i \leq l$ ) is repeated a total of  $2^p + 2^q$  times in the matrix  $[M_1 \ M_2]^T$ .

Proposition 1 has also been used by [12] in the construction of resilient S-boxes. However we improve upon the construction of [12] by utilizing the following two ideas.

1. We use all the  $2^m - 1$  rows of the matrix  $L(C)$ . In contrast, [12] uses at most  $2^{m-1}$  rows of  $L(C)$ .
2. We allow a row of  $L(C)$  to be repeated  $2^{r_1}$  or  $2^{r_1} + 2^{r_2}$  or  $2^{r_1} + 2^{r_2} + 2^{r_3}$  times as required. On the other hand, the number of times a row of  $L(C)$  can be repeated in [12] is of the form  $2^r$ .

It turns out that a proper utilization of the above two techniques result in significant improvement in nonlinearity. We will require  $(r, m)$  S-boxes with very high nonlinearity. For this we propose to use the best known results which we summarize in the following definition.

**Definition 1.** *Let  $G$  be an  $(r, m)$  S-box satisfying the following.*

1. If  $r < m$ ,  $G$  is a constant S-Box.
2. If  $m \leq r < 2m$ ,  $G$  is a maximally nonlinear S-Box [6].
3. If  $r \geq 2m$  and  $r$  is even,  $G$  is a perfect nonlinear S-Box [11].
4. If  $r \geq 2m$  and  $r$  is odd,  $G$  is concatenation of two perfect nonlinear S-

*Boxes (see Section 2.2).*

*Then we say that  $G$  is a PROPER S-box.*

The following result summarizes the best known results on the nonlinearity of PROPER S-boxes.

**Proposition 2.** *Let  $G$  be an  $(r, m)$  PROPER S-box. Then*

1. *If  $r < m$ ,  $nl(G) = 0$ .*
2. *If  $m \leq r < 2m$ , then  $nl(G) = 2^{r-1} - 2^{\frac{r-1}{2}}$  if  $r$  is odd and  $nl(G) \geq 2^{r-1} - 2^{\frac{r}{2}}$  if  $r$  is even.*
3. *If  $r \geq 2m$ , then  $nl(G) = 2^{r-1} - 2^{\frac{r}{2}-1}$  if  $r$  is even and  $nl(G) = 2^{r-1} - 2^{\frac{r-1}{2}}$  if  $r$  is odd.*

Now we are in a position to describe a new construction of resilient S-boxes. The construction has two parts. In Part-A, we compute the number of rows of  $L(C)$  to be used and the number of times each row is to be repeated. The output of Part-A is a *list* of the form  $list = \langle (n_1, R_1), (n_2, R_2), \dots, (n_k, R_k) \rangle$  which signifies that  $n_i$  rows of  $L(C)$  are to be repeated  $R_i$  times each. Part-A also computes a variable called **effect** which determines the nonlinearity of the S-box (see Theorem 8). In Part-B of the construction, we choose PROPER functions based on *list* and describe the actual construction of the S-box.

## Construction-I

1. Input: Positive integers  $(n, m)$  and  $t$ .
2. Output: A nonlinear  $(n, m, t)$ -resilient S-box  $F$ .

### Part-A

1. Obtain minimum  $u$  such that  $[u, m, t+1]$  code  $C$  exists.
2. Case:  $n - u \leq 0$ , then function cannot be constructed using this method. Hence stop.
3. Case:  $n - u \geq 0$ 
  - (a)  $0 \leq n - u < m$ ;  $list = \langle (2^{n-u}, 1) \rangle$  and **effect** = 1.
  - (b)  $m \leq n - u < 2m - 1$ ;  $list = \langle (2^{m-1}, 2^{n-u-m+1}) \rangle$  and **effect** =  $2^{n-u-m+1}$ .
  - (c)  $n - u = 2m - 1$ ;  $list = \langle (2^{m-1}, 2^m) \rangle$  and **effect** =  $2^{\lfloor \frac{m}{2} \rfloor + 1}$ .
  - (d)  $2m \leq n - u < 3m$ .
    - (i)  $n - u = 2m + 2e$ ;  $m$  even;  $0 \leq e < \frac{m}{2}$ ;  
 $list = \langle (1, 2^{m+2e+1}), (2^m - 2, 2^{m+2e}) \rangle$  and **effect** =  $2^{e+1+\frac{m}{2}}$ .
    - (ii)  $n - u = 2m + 2e + 1$ ;  $m$  even;  $0 \leq e \leq \frac{m}{2} - 1$ ;  
      - $0 \leq e \leq \frac{m}{2} - 2$ ;  
 $list = \langle (2, 2^{m+2e+1} + 2^{2e+1} + 2^{2e}), (2^m - 3, 2^{m+2e+1} + 2^{2e+1}) \rangle$   
and **effect** =  $2^{2e+1} + 2^{2e} + 2^{e+1+\frac{m}{2}}$ .
      - $e = \frac{m}{2} - 1$ ;  $list = \langle (2^{m-1}, 2^m) \rangle$  and **effect** =  $2^m$ .
    - (iii)  $n - u = 2m + 2e + 1$ ;  $m$  odd;  $0 \leq e \leq \lfloor \frac{m}{2} \rfloor - 1$ ;  
 $list = \langle (1, 2^{m+2e+2}), (2^m - 2, 2^{m+2e+1}) \rangle$  and **effect** =  $2^{\frac{m+2e+3}{2}}$ .
    - (iv)  $n - u = 2m + 2e$ ;  $m$  odd;  $0 \leq e < \lfloor \frac{m}{2} \rfloor$ ;



- $list = \langle (2^m - 2, 2^{m+2e} + 2^{2e+1}), (1, 2^{2e+2}) \rangle$   
 and  $effect = 2^{2e+1} + 2^{\frac{m+2e+1}{2}}$ .
- (v)  $n - u = 3m - 1$ ;  $m$  odd;  
 $list = \langle (2^{m-1}, 2^{2m}) \rangle$  and  $effect = 2^m$ .
- (e)  $n - u \geq 3m$ .
  - (i)  $n - u = 3m + 2e + 1$ ;  $e \geq 0$ ;  
 $list = \langle (2^{m-1}, 2^{2m+2e+2}) \rangle$  and  $effect = 2^{m+e+1}$ .
  - (ii)  $n - u = 3m + 2e$ ; ( $m$  even;  $e \geq \frac{m}{2}$ ) or ( $m$  odd;  $0 \leq e < \lfloor \frac{m}{2} \rfloor$ );  
 $list = \langle (2, 2^{2m+2e} + 2^{m+2e} + 2^{m+2e-1}), (2^m - 3, 2^{2m+2e} + 2^{m+2e}) \rangle$   
 and  $effect = 2^{m+e} + 2^{e+1+\frac{m}{2}}$ .
  - (iii)  $n - u = 3m + 2e$ ;  $m$  even;  $0 \leq e < \frac{m}{2}$   
 $list = \langle (2^m - 2, 2^{2m+2e} + 2^{m+2e+1}), (1, 2^{m+2e+2}) \rangle$   
 and  $effect = 2^{m+e} + 2^{e+1+\frac{m}{2}}$ .
  - (iv)  $n - u = 3m + 2e$ ;  $m$  odd;  $e \geq \lfloor \frac{m}{2} \rfloor$   
 $list = \langle (2^m - 2, 2^{2m+2e} + 2^{m+2e+1}), (1, 2^{m+2e+2}) \rangle$   
 and  $effect = 2^{m+e} + 2^{e+\frac{m+1}{2}}$ .

## Part-B

1. If  $list = \langle (2^s, 2^r) \rangle$ ;
  - Obtain  $L(C, 1, 2^s)$  from  $L(C)$  by selecting first  $2^s$  rows of  $L(C)$ .
  - Let  $G$  be an  $(r, m)$  PROPER S-box.
  - Define  $F = G \oplus L(C, 1, 2^s)$ .
  - This covers cases 3.(a),(b),(c),(d)(ii) second item, (d)(v) and e(i) of Part-A.
2. Case: 3(d)(i) of Part-A
  - Let  $G_1$  and  $G_2$  be  $(m + 2e + 1, m)$  and  $(m + 2e, m)$  PROPER S-boxes.
  - Define  $F_1 = G_1 \oplus L(C, 1, 1)$ ,  $F_2 = G_2 \oplus L(C, 2, 2^m - 1)$ .
  - $F$  is the concatenation of  $F_1$  and  $F_2$ .
3. Case: 3(d)(ii) first item of Part-A and  $e = 0$ 
  - Let  $G_1$  and  $G_2$  be  $(m + 1, m)$  and  $(1, m)$  PROPER S-boxes.
  - Define  $F_1 = G_1 \oplus L(C)$ ,  $F_2 = G_2 \oplus L(C)$ ,  $F_3 = L(C, 1, 2)$ .
  - $F$  is the concatenation of  $F_1, F_2$  and  $F_3$ .
4. Case: 3(d)(ii) first item of Part-A and  $e \neq 0$ 
  - Let  $G_1, G_2$  and  $G_3$  be  $(m + 2e + 1, m)$ ,  $(2e + 1, m)$  and  $(2e, m)$  PROPER S-boxes.
  - Define  $F_1 = G_1 \oplus L(C)$ ,  $F_2 = G_2 \oplus L(C)$ ,  $F_3 = G_3 \oplus L(C, 1, 2)$ .
  - $F$  is the concatenation of  $F_1, F_2$  and  $F_3$ .
5. Case: 3(d)(iii) of Part-A
  - Let  $G_1$  and  $G_2$  be  $(m + 2e + 2, m)$  and  $(m + 2e + 1, m)$  PROPER S-boxes.
  - Define  $F_1 = G_1 \oplus L(C, 1, 1)$ ,  $F_2 = G_2 \oplus L(C, 2, 2^m - 1)$ .
  - $F$  is the concatenation of  $F_1$  and  $F_2$ .
6. Case: 3(d)(iv) of Part-A
  - Let  $G_1, G_2$  and  $G_3$  be  $(m + 2e, m)$ ,  $(2e + 2, m)$  and  $(2e + 1, m)$  PROPER S-boxes.
  - Define  $F_1 = G_1 \oplus L(C, 1, 2^m - 2)$ ,  $F_2 = G_2 \oplus L(C, 2^m - 1, 2^m - 1)$ ,

$$F_3 = G_3 \oplus L(C, 1, 2^m - 2) .$$

- $F$  is the concatenation of  $F_1, F_2$  and  $F_3$  .

7. Case: 3(e)(ii) of Part-A

- Let  $G_1, G_2$  and  $G_3$  be  $(2m + 2e, m)$ ,  $(m + 2e, m)$  and  $(m + 2e - 1, m)$  PROPER S-boxes.
- Define  $F_1 = G_1 \oplus L(C)$ ,  $F_2 = G_2 \oplus L(C)$ ,  $F_3 = G_3 \oplus L(C, 1, 2)$  .
- $F$  is the concatenation of  $F_1, F_2$  and  $F_3$  .

8. Case: 3(e)(iii) and 3(e)(iv) of Part-A

- Let  $G_1, G_2$  and  $G_3$  be  $(2m + 2e, m)$ ,  $(m + 2e + 2, m)$  and  $(m + 2e + 1, m)$  PROPER S-boxes.
- Define  $F_1 = G_1 \oplus L(C, 1, 2^m - 2)$ ,  $F_2 = G_2 \oplus L(C, 2^m - 1, 2^m - 1)$ ,  $F_3 = G_3 \oplus L(C, 1, 2^m - 2)$  .
- $F$  is the concatenation of  $F_1, F_2$  and  $F_3$  .

**Theorem 8.** *Construction-I provides a nonlinear  $(n, m, t)$ -resilient S-box with nonlinearity  $= (2^{n-1} - 2^{u-1} \times \text{effect})$ , where effect is as computed in Part-A.*

*Proof.* There are several things to be proved.

(a) The output function  $F$  is an  $(n, m)$  S-box. (b)  $F$  is  $t$ -resilient. (c)  $nl(f) = (2^{n-1} - 2^{u-1} \times \text{effect})$ .

*Proof of (a)* The output of Part-A is a list  $= \langle (n_1, R_1), (n_2, R_2), \dots, (n_k, R_k) \rangle$ . Part-B ensures that for  $1 \leq i \leq k$ ,  $n_i$  rows of  $L(C)$  are repeated  $R_i$  times each. It is easy to verify that in each case of Part-A we have  $\sum_{i=1}^k n_i R_i = 2^{n-u}$ . Since each row  $L_{i,*}$  of  $L(C)$  defines a  $(u, m)$  S-box, ultimately  $F$  is an  $(n, m)$  S-box.

*Proof of (b)* Each row  $L_{i,*}$  of  $L(C)$  defines a  $t$ -resilient  $(u, m)$  S-box.  $F$  is formed by concatenating the rows of  $L(C)$  one or more times. Hence  $F$  is  $t$ -resilient.

*Proof of (c)* The nonlinearity calculation is similar for all the cases. As an example, we perform the calculation for Case 3(e)(ii). In this case, Part-A computes  $list = \langle (2, 2^{2m+2e} + 2^{m+2e} + 2^{m+2e-1}), (2^m - 3, 2^{2m+2e} + 2^{m+2e}) \rangle$ . Let  $R_1 = 2^{2m+2e} + 2^{m+2e} + 2^{m+2e-1}$  and  $R_2 = 2^{2m+2e} + 2^{m+2e}$ . Rows  $L_{1,*}$  and  $L_{2,*}$  of  $L(C)$  are repeated  $R_1$  times each and each of the rows  $L_{3,*}$  to  $L_{2^m-1,*}$  is repeated  $R_2$  times each. Part-B uses three PROPER functions  $G_1, G_2$  and  $G_3$  to construct S-boxes  $F_1, F_2$  and  $F_3$  respectively.  $F$  is the concatenation of  $F_1, F_2$  and  $F_3$ . We have to show that if  $\nu$  is a non constant  $m$ -variable linear function and  $\lambda$  is an  $n$ -variable linear function, then  $d(\nu \circ F, \lambda) \geq (2^{n-1} - 2^{u-1} \times \text{effect})$ . We write  $\lambda$  as  $\lambda(y_1, \dots, y_{n-u}, x_1, \dots, x_u) = \lambda_1(y_1, \dots, y_{n-u}) \oplus \lambda_2(x_1, \dots, x_u)$ . Let  $\nu(z_1, \dots, z_m) = \langle (c_1, \dots, c_m), (z_1, \dots, z_m) \rangle$  for some non-zero vector  $c = (c_1, \dots, c_m) \in F_2^m$ . The Boolean function  $\nu \circ F$  is a concatenation of Boolean functions  $\nu \circ F_1$ ,  $\nu \circ F_2$  and  $\nu \circ F_3$ . For  $1 \leq i \leq 2$ ,  $\nu \circ F_i = (\nu \circ G_i) \oplus (L(C)c^T)$  and  $\nu \circ F_3 = (\nu \circ G_3) \oplus (L(C, 1, 2)c^T)$ . Using Proposition 1, we know that all the entries of the column vector  $L(C)c^T$  are distinct  $u$ -variable linear functions. Let  $L(C)c^T = [\mu_1, \dots, \mu_{2^m-1}]^T$ . The function  $\nu \circ F$  is a concatenation of the  $\mu_i$ 's and their complements. Further,  $\mu_1$  and  $\mu_2$  are repeated  $R_1$  times and  $\mu_3, \dots, \mu_{2^m-1}$  are repeated  $R_2$  times in the construction of  $\nu \circ F$ . If  $\lambda \notin \{\mu_1, \dots, \mu_{2^m-1}\}$  then  $d(\lambda_2, \mu_i) = 2^{u-1}$  for each  $1 \leq i \leq 2^m - 1$  and hence  $d(\nu \circ F, \lambda) = 2^{n-u}(2^{u-1}) = 2^{n-1}$ . Now suppose  $\lambda_2 = \mu_i$  for some  $i \in \{1, \dots, 2^m - 1\}$ . In this case  $d(\nu \circ F, \lambda)$

will be less than  $2^{n-1}$  and the actual value is determined by the repetition factors  $R_1$  and  $R_2$ . There are two cases to consider.

*Case 1:*  $\lambda_2 = \mu_1$  or  $\mu_2$ . Without loss of generality we assume  $\lambda_2 = \mu_1$ , the other case being similar. Since  $\lambda_2 = \mu_1$ , we have  $d(\lambda_2, \mu_i) = 2^{u-1}$  for  $2 \leq i \leq 2^m - 1$ . The function  $\mu_2$  is repeated  $R_1$  times and each of the functions  $\mu_3, \dots, \mu_{2^m-1}$  is repeated  $R_2$  times. So the total contribution of  $\mu_2, \mu_3, \dots, \mu_{2^m-1}$  to  $d(\nu \circ F, \lambda)$  is  $2^{u-1}(R_1 + (2^m - 3)R_2)$ . We now have to compute the contribution of  $\mu_1$  to  $d(\nu \circ F, \lambda)$ . The function  $\mu_1$  is repeated in  $\nu \circ F_i$  by XORing with  $\nu \circ G_i$ . Hence the contribution of  $\mu_1$  to  $d(F, \lambda)$  is equal to  $2^u(nl(\nu \circ G_1) + nl(\nu \circ G_2) + nl(\nu \circ G_3)) = 2^u(nl(G_1) + nl(G_2) + nl(G_3))$  since  $nl(\nu \circ G_i) = nl(G_i)$ . Each  $G_i$  is a PROPER function whose nonlinearity is given by Proposition 2.

Hence,  $d(\nu \circ F, \lambda) = 2^{u-1}(R_1 + (2^m - 3)R_2 + 2(nl(G_1) + nl(G_2) + nl(G_3))) = 2^{u-1}(2^{n-u} - (R_1 - 2(nl(G_1) + nl(G_2) + nl(G_3)))) = 2^{n-1} - 2^{u-1}(R_1 - 2(nl(G_1) + nl(G_2) + nl(G_3)))$ .

From the given conditions, it is easy to verify that  $\text{effect} = R_1 - 2(nl(G_1) + nl(G_2) + nl(G_3))$  and so  $d(\nu \circ F, \lambda) = (2^{n-1} - 2^{u-1} \times \text{effect})$ .

*Case 2:*  $\lambda_2 = \mu_i$  for some  $i \in \{3, \dots, 2^m - 1\}$ . In this case we proceed as in the previous case to obtain  $d(\nu \circ F, \lambda) = 2^{u-1}(2R_1 + (2^m - 4)R_2) + 2^u(nl(G_1) + nl(G_2)) = 2^{u-1}(2R_1 + (2^m - 4)R_2 + 2(nl(G_1) + nl(G_2))) = 2^{u-1}(2^{n-u} - R_2 + 2(nl(G_1) + nl(G_2))) = 2^{n-1} - 2^{u-1}(R_2 - 2(nl(G_1) + nl(G_2))) > 2^{n-1} - 2^{u-1} \times \text{effect}$ , since  $\text{effect} = R_1 - 2(nl(G_1) + nl(G_2) + nl(G_3)) > R_1 - 2(nl(G_1) + nl(G_2))$ .

By *Case 1* and *Case 2* above it follows that  $nl(\nu \circ F) = 2^{n-1} - 2^{u-1} \times \text{effect}$ . Hence  $nl(F) = 2^{n-1} - 2^{u-1} \times \text{effect}$ .  $\square$

## 6 Results and Comparisons

Here we compare the construction methods described in this paper to the known construction methods.

### 6.1 Degree Comparison Based on MZZ Construction

We present examples to show the advantage of the MZZ method over the Cheon method. Cheon method cannot construct  $(n, m, t)$ -resilient function of degree  $d \geq m \geq 2$  if the following two conditions hold.

(1) 

$t$	1	2 to 3	4 to 7	8 to 15	16 to 31
$m$	$m \geq 1$	$m \geq 2$	$m \geq 3$	$m \geq 4$	$m \geq 5$

(2) The parameters  $n, d + 1, t + 1$  satisfy Griesmer bound with equality.

We next present some examples of  $n, m, d$  and  $t$  satisfying condition (1) and (2) such that the MZZ method can be used to construct  $(n, m, t)$ -resilient function with degree  $d$ .

(a)  $t = 1, 2 \leq m \leq d, n = d + 2$ . It is easy to check that a  $[d + 2, d + 1, 2]$  code exists.

(b)  $t = 2, 2 \leq m \leq d, (n, d) = (6, 2), (7, 3), (8, 4), (9, 5), (10, 6), (11, 7)$ . In each case an  $[n, d + 1, t + 1]$  code exists.

**Table 1.** Comparison of nonlinearity obtained by MZZ Construction to that obtained by Cheon [4].

Function	(10, 3, 1, 5)	(18, 4, 2, 10)	(24, 5, 2, 15)	(24, 7, 3, 12)	(28, 6, 4, 14)
Cheon [4, Theorem 5]	8	$2^{16} + 2^9$	$2^{23} - 2^{20} + 2^7$	$2^{10}$	$2^{12}$
MZZ	$2^9 - 2^7$	$2^{17} - 2^{12}$	$2^{23} - 2^{16}$	$2^{23} - 2^{17}$	$2^{27} - 2^{20}$

(c)  $t = 3$ ,  $2 \leq m \leq d$ ,  $(n, d) = (7, 2), (8, 3), (11, 6), (12, 7), (13, 8)$ . In each case an  $[n, d + 1, t + 1]$  code exists.

In (a) to (c) above an  $(n, m, t)$ -resilient function with degree  $d$  can be constructed using MZZ method, but cannot be constructed using Cheon method (see Theorem 5). Now we present some examples where both MZZ and Cheon method construct  $(n, m, t)$ -resilient function with degree  $d$  and compare their nonlinearity using Theorem 6. An  $(n, m, d, t)$  S-box is an  $(n, m, t)$ -resilient S-box with degree  $d$ .

We see that in each case the nonlinearity obtained by the MZZ method is far superior to that obtained by the Cheon method.

## 6.2 Nonlinearity Comparison Based on Construction-I

We compare the nonlinearity obtained by Construction-I to the nonlinearity obtained in Theorem 4 of [12]. The nonlinearity obtained in [12] is better than the nonlinearity obtained by other methods. Hence we do not compare our method with the other methods. It is to be noted that in certain cases the search technique of [7] provides better nonlinearity than [12].

Our first observation is that *the nonlinearity obtained by Construction-I is at least as large the nonlinearity obtained in [12]*. The intuitive reason is that we use all the rows of the matrix  $L(C)$  and hence the repetition factor is less than that of [12]. The detailed verification of the superiority of Construction-I over [12] is straightforward but tedious. In the next table we summarize the cases under which Construction-I yields higher nonlinearity than [12]. We list the different cases of Part-A corresponding to the different rows of the table.

**Table 2.** Comparison of Construction-I nonlinearity with the nonlinearity of [12].

Case	Nonlinearity of [12]	Construction-I nonlinearity
$2m \leq n - u < 3m - 3$ , $\pi$ even	$2^{n-1} - 2^{(n+u-m+1)/2}$	$2^{n-1} - 2^{(n+u-m-1)/2} - 3 \times 2^{n-2m-2}$ (1)
		$2^{n-1} - 2^{(n+u-m-1)/2} - 2^{n-2m}$ (2)
$2m \leq n - u < 3m - 3$ , $\pi$ odd	$2^{n-1} - 2^{(n+u-m+2)/2}$	$2^{n-1} - 2^{(n+u-m)/2}$ (3)
$n - u = 3m - 3$	$2^{n-1} - 2^{(u+m-1)}$	$2^{n-1} - \frac{11}{16} 2^{(u+m-1)}$ (4)
$n - u \geq 3m$ , $\pi$ odd	$2^{n-1} - 2^{(n+u-m)/2}$	$2^{n-1} - 2^{(n+u-m)/2} (\frac{1}{2} + \frac{1}{2^{m/2}})$ (5)
		$2^{n-1} - 2^{(n+u-m)/2} (\frac{1}{2} + \frac{1}{2^{((m+1)/2)}})$ (6)

(1) Case 3(d)(ii)first item; (2) Case 3(d)(iv); (3) Case 3(d)(i) and Case 3(d)(iii); (4) Case 3(d)(ii)first item; (5) Case 3(e)(iii),  $m > 2$  and Case 3(e)(ii),  $m > 2$ ; (6) Case 3(e)(iv),  $m > 1$ .

In Tables 3 to 5 we provide some concrete examples of cases where the nonlinearity obtained by Construction-I is better than that obtained by [12]. Each entry of Tables 3 to 5 is of the form  $(a, b)$ , where  $a$  is the nonlinearity obtained by [12] and  $b$  is the nonlinearity obtained by Construction-I.

The linear codes used in Table 3 are  $[5, 4, 2]$ ,  $[7, 4, 3]$  and  $[8, 4, 4]$ . The 2nd, 4th, and 6th rows give the nonlinearity of  $(n, m, t)$ -resilient functions corresponding to the codes  $[5, 4, 2]$ ,  $[7, 4, 3]$  and  $[8, 4, 4]$  respectively for different values of  $n$ .

The linear codes used in Table 4 are  $[6, 5, 2]$ ,  $[9, 5, 3]$  and  $[10, 5, 4]$ .

The linear codes used in Table 5 are  $[7, 6, 2]$ ,  $[10, 6, 3]$  and  $[10, 6, 4]$ .

Nonlinearity of  $(36, 8, t)$  resilient S-box has been used as very important examples in [8, 7, 12]. Now we compare our nonlinearity with those.

The results of [7] are not constructive. They show that resilient S-box with such parameter exist. *Note that, except for resiliencies of order 1 and 3 our nonlinearity is better than nonlinearity of [12].* It should also be noted that in all the cases we provide construction with currently best known nonlinearity.

**Table 3.** Comparison of Construction-I nonlinearity with [12] for  $m = 4$  and resiliency = 1, 2, 3.

$n = 13$ ( $2^{12} - 2^8$ ), ( $2^{12} - 2^7$ )	$n = 14$ ( $2^{13} - 2^8$ ), ( $2^{13} - \frac{11}{16}2^8$ )	$n = 17$ ( $2^{16} - 2^9$ ), ( $2^{16} - \frac{3}{4}2^9$ )	$n = 19$ ( $2^{18} - 2^{10}$ ), ( $2^{18} - \frac{3}{4}2^{10}$ )
$n = 15$ ( $2^{14} - 2^{10}$ ), ( $2^{14} - 2^9$ )	$n = 16$ ( $2^{15} - 2^{10}$ ), ( $2^{15} - \frac{11}{16}2^{10}$ )	$n = 19$ ( $2^{18} - 2^{11}$ ), ( $2^{18} - \frac{3}{4}2^{11}$ )	$n = 21$ ( $2^{20} - 2^{12}$ ), ( $2^{20} - \frac{3}{4}2^{12}$ )
$n = 16$ ( $2^{15} - 2^{11}$ ), ( $2^{15} - 2^{10}$ )	$n = 17$ ( $2^{16} - 2^{11}$ ), ( $2^{16} - \frac{11}{16}2^{11}$ )	$n = 20$ ( $2^{19} - 2^{12}$ ), ( $2^{19} - \frac{3}{4}2^{12}$ )	$n = 22$ ( $2^{21} - 2^{13}$ ), ( $2^{21} - \frac{3}{4}2^{13}$ )

**Table 4.** Comparison of Construction-I nonlinearity with [12] for  $m = 5$  and resiliency = 1, 2, 3.

$n = 16$ ( $2^{15} - 2^9$ ), ( $2^{15} - \frac{5}{8}2^9$ )	$n = 17$ ( $2^{16} - 2^{10}$ ), ( $2^{16} - 2^9$ )	$n = 18$ ( $2^{17} - 2^{10}$ ), ( $2^{17} - \frac{11}{16}2^{10}$ )	$n = 21$ ( $2^{20} - 2^{11}$ ), ( $2^{20} - \frac{5}{8}2^{11}$ )
$n = 19$ ( $2^{18} - 2^{12}$ ), ( $2^{18} - \frac{5}{8}2^{12}$ )	$n = 20$ ( $2^{19} - 2^{13}$ ), ( $2^{19} - 2^{12}$ )	$n = 21$ ( $2^{20} - 2^{13}$ ), ( $2^{20} - \frac{11}{16}2^{13}$ )	$n = 24$ ( $2^{23} - 2^{14}$ ), ( $2^{23} - \frac{5}{8}2^{14}$ )
$n = 18$ ( $2^{17} - 2^{11}$ ), ( $2^{17} - \frac{5}{8}2^{11}$ )	$n = 19$ ( $2^{18} - 2^{12}$ ), ( $2^{18} - 2^{11}$ )	$n = 20$ ( $2^{19} - 2^{12}$ ), ( $2^{19} - \frac{11}{16}2^{12}$ )	$n = 25$ ( $2^{24} - 2^{15}$ ), ( $2^{24} - \frac{5}{8}2^{15}$ )

**Table 5.** Comparison of Construction-I nonlinearity with [12] for  $m = 6$  and resiliency = 1, 2, 3.

$n = 19$ ( $2^{18} - 2^{11}$ ), ( $2^{18} - 2^{10}$ )	$n = 20$ ( $2^{19} - 2^{11}$ ), ( $2^{19} - \frac{19}{32}2^{11}$ )	$n = 21$ ( $2^{20} - 2^{12}$ ), ( $2^{20} - 2^{11}$ )	$n = 22$ ( $2^{21} - 2^{12}$ ), ( $2^{21} - \frac{11}{16}2^{12}$ )
$n = 22$ ( $2^{21} - 2^{14}$ ), ( $2^{21} - 2^{13}$ )	$n = 23$ ( $2^{22} - 2^{14}$ ), ( $2^{22} - \frac{19}{32}2^{14}$ )	$n = 24$ ( $2^{23} - 2^{15}$ ), ( $2^{23} - 2^{14}$ )	$n = 25$ ( $2^{24} - 2^{15}$ ), ( $2^{24} - \frac{11}{16}2^{15}$ )
$n = 22$ ( $2^{21} - 2^{14}$ ), ( $2^{21} - 2^{13}$ )	$n = 23$ ( $2^{22} - 2^{14}$ ), ( $2^{22} - \frac{19}{32}2^{14}$ )	$n = 24$ ( $2^{23} - 2^{15}$ ), ( $2^{23} - 2^{14}$ )	$n = 25$ ( $2^{24} - 2^{15}$ ), ( $2^{24} - \frac{11}{16}2^{15}$ )

**Table 6.** Comparison of nonlinearity of  $(36, 8, t)$ -resilient S-boxes using different methods.

$t$	7	6	5	4	3	2	1
[8]	$2^{35} - 2^{27}$	$2^{35} - 2^{27}$	$2^{35} - 2^{26}$	$2^{35} - 2^{25}$	$2^{35} - 2^{24}$	$2^{35} - 2^{23}$	$2^{35} - 2^{22}$
[7]	$2^{35} - 2^{22}$	-	$2^{35} - 2^{23}$	$2^{35} - 2^{22}$	$2^{35} - 2^{22}$	$2^{35} - 2^{21}$	$2^{35} - 2^{21}$
[12]	$2^{35} - 2^{25}$	$2^{35} - 2^{24}$	$2^{35} - 2^{23}$	$2^{35} - 2^{23}$	$2^{35} - 2^{20}$	$2^{35} - 2^{20}$	$2^{35} - 2^{18}$
Ours	$2^{35} - 2^{24}$	$2^{35} - \frac{35}{64}2^{24}$	$2^{35} - \frac{19}{32}2^{23}$	$2^{35} - 2^{22}$	$2^{35} - 2^{20}$	$2^{35} - \frac{9}{16}2^{20}$	$2^{35} - 2^{18}$
Codes	[20, 8, 8]	[19, 8, 7]	[17, 8, 6]	[16, 8, 5]	[13, 8, 4]	[12, 8, 3]	[9, 8, 2]

## 7 Conclusion

In this paper we consider the construction of nonlinear resilient S-boxes. We prove that the correlation immunity of a resilient S-box is preserved under composition with an arbitrary Boolean function. Our main contribution is to obtain two construction methods for nonlinear resilient S-boxes. The first construction is a simple modification of an elegant construction due to Zhang and Zheng [20]. This provides  $(n, m, t)$ -resilient S-boxes with degree  $d > m$ . We *prove* that the modified Zhang Zheng construction is superior to the only previously known construction [4] which provided degree  $d > m$ . Our second construction is based on concatenation of small affine function to build nonlinear resilient S-boxes. We sharpen the technique to construct  $(n, m, t)$ -resilient S-boxes with the currently best known nonlinearity.

## References

1. C. Bennett, G. Brassard and J. Robert. Privacy Amplification by Public Discussion. *SIAM Journal of Computing*, volume 17, pages 210–229, 1988.
2. P. Camion, C. Carlet, P. Charpin and N. Sendrier . On correlation immune functions. In *Advances in Cryptology – CRYPTO 1991*, pages 86–100, Lecture Notes in Computer Science, Springer-Verlag, 1992.
3. S. Chee, S. Lee, D. Lee and S. H. Sung . On the correlation immune functions and their nonlinearity. In *Advances in Cryptology – Asiacrypt 1996*, pages 232–243, Lecture Notes in Computer Science, Springer-Verlag, 1996.
4. Jung Hee Cheon. Nonlinear Vector Resilient Functions. In *Advances in Cryptology – CRYPTO 2001*, pages 458–469, Lecture Notes in Computer Science, Springer-Verlag, 2001.
5. B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich and R. Smolensky. The Bit Extraction Problem or  $t$ -resilient Functions. *IEEE Symposium on Foundations of Computer Science*, volume 26, pages 396–407, 1985.
6. Hans Dobbertin, Almost Perfect Nonlinear Power Functions on  $GF(2^n)$ : The Welch Case. *IEEE Transactions on Information Theory*, Vol 45 , No 4, pp. 1271–1275 , 1999.

7. T. Johansson and E. Pasalic. A construction of resilient functions with high non-linearity. *International Symposium on Information Theory*, 2000.
8. K. Kurosawa, T. Satoh and K. Yamamoto. Highly nonlinear  $t$ -resilient functions . *Journal of Universal Computer Science*, vol.3, no. 6, pp. 721-729, Springer Publishing Company, 1997.
9. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North Holland, 1977.
10. K. Nyberg. Perfect Nonlinear S-boxes. In *Advances in Cryptology – EUROCRYPT 1991*, pages 378–386, Lecture Notes in Computer Science, Springer-Verlag, 1991.
11. K. Nyberg. Differentially uniform mapping for cryptography. In *Advances in Cryptology – EUROCRYPT 1993*, pages 55–65, Lecture Notes in Computer Science, Springer-Verlag, 1994.
12. E. Pasalic and S. Maitra. Linear Codes in Generalized Construction of Resilient Functions with Very High Nonlinearity. *Earlier version in SAC 2001. To appear in IEEE Transactions on Information Theory*.
13. B. Preneel. Analysis and design of cryptographic hash functions, doctoral dissertation, K.U. Leuven, 1993.
14. O. S. Rothaus. On bent functions. *Journal of Combinatorial Theory, Series A*, 20:300–305, 1976.
15. P. Sarkar and S. Maitra. Construction of Nonlinear Boolean Functions with Important Cryptographic Properties. In *Advances in Cryptology – EUROCRYPT 2000*, pages 485–506, Lecture Notes in Computer Science, Springer-Verlag, 2000.
16. J. Seberry, X.-M. Zhang and Y. Zheng . On construction and nonlinearity of correlation immune Boolean functions. In *Advances in Cryptology – EUROCRYPT 1993*, pages 181–199, Lecture Notes in Computer Science, Springer-Verlag, 1994.
17. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776–780, September 1984.
18. D. R. Stinson and J. L. Massey. An Infinite Class of Counterexamples to a Conjecture Concerning Nonlinear Resilient Functions. *Journal of Cryptology*, volume 8, pages 167–173, 1995.
19. G. Xiao and J. L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Transactions on Information Theory*, pages 569–571, 1988.
20. X.-M. Zhang and Y. Zheng, On Cryptographically Resilient Functions. *IEEE Transactions on Information Theory*, Vol 43 , No 5, pp. 1740-1747 , 1997.

# An Upper Bound on the Number of $m$ -Resilient Boolean Functions

Claude Carlet<sup>1</sup> and Aline Gouget<sup>2</sup>

<sup>1</sup> INRIA, Domaine de Voluceau, BP 105 – 78153 Le Chesnay Cedex, France,  
also member of GREYC-Caen and of University of Paris 8,  
`Claude.Carlet@inria.fr`

<sup>2</sup> GREYC, Université de Caen, 14032 Caen Cedex, France,  
`gouget@info.unicaen.fr`

**Abstract.** The enumeration of  $m$ -resilient Boolean functions in  $n$  variables would be a quite useful information for cryptography. But it seems to be an intractable open problem. Upper and lower bounds have appeared in the literature in the mid 80's. Since then, improving them has been the goal of several papers. In this paper, we give a new upper bound which partially improves upon all the known bounds.

**Keywords:** Cryptography, Stream cipher, Boolean function, Resilient function

## 1 Introduction

The principle of private cryptography relies on the share-out of a private key between the sender of a message and its receiver. Symmetric cryptosystems are commonly used owing to their efficiency. Currently, there is no mathematical proof to ensure the unconditional security of the system except for the famous Vernam [13] scheme. This system produces the encoded text by adding bitwisely the plain text and the private key. Then the receiver retrieves the plain text by using the same addition of the encoded text and the private key. In practice, since the length of the private key must equal the length of the plain text, pseudo-random generators are used for stream ciphers in order to minimize the size of the private key (but the unconditional security is then no longer ensured). In order to achieve maximal security, these systems are much studied.

The basic component of a keystream generator is the Linear Feedback Shift Register (LFSR). The generic example of a keystream generator is composed of  $n$  LFSR whose outputs are combined by a Boolean function from  $\mathbb{F}_2^n$  to  $\mathbb{F}_2$ . The security of the system relies, in a central way, on the choice of the Boolean function. Subsequently, the Boolean functions used to combine several LFSR, called combining functions, must fulfil several criteria. They must be *balanced*, i.e., they must take the value 1 and the value 0 with the same probability on the set  $\mathbb{F}_2^n$ . They must have high algebraic degrees (see definition at section 2) so that the keystream generator resists the Berlekamp-Massey's attack [6]. The generator must also resist the Siegenthaler's correlation attack [12]. This comes



down to choose a combining function which is correlation-immune of a high order  $m$  [11], *i.e.*, whose output distribution does not change when  $m$  input values (*i.e.*,  $m$  coordinates of the input vector) are fixed. If the combining function is correlation-immune of order  $m$ , the attacker has to guess the initialization of at least  $m + 1$  LFSR to observe a correlation between them and the output of the pseudo-random generator during a correlation attack. Combining functions must also have high non-linearities in order to prevent linear approximation. Of course these criteria are partially opponent and tradeoffs exist.

Enumerating the Boolean functions satisfying one or several of these criteria is useful for several reasons. Firstly because it indicates for which values of the parameters  $(n, \dots)$  there is a chance of finding good cryptographic functions by random search. Secondly because a large number of functions is necessary if we want to impose extra constraints on the functions or if we want to modify the cryptosystems using them by having the function as part of the secret key.

Mitchell [7] proposed a number of open problems with partial results about enumerating Boolean functions satisfying various criteria, including balancedness and correlation-immunity. The first bounds on the number of first order correlation-immune Boolean functions were lower bounds (see [7,14,8,5]). In 1990, Yang and Guo published the first upper bound on such functions. Park, Lee, Sung and Kim [8] proceeded further and improved upon Yang-Guo's bound. In 1995, Schneider [10] used a new idea to improve upon previous bounds. He obtained bounds for the numbers of  $m$ th-order correlation-immune functions and of  $m$ -resilient functions. Carlet and Klapper [1] obtained a general upper bound on the number of Boolean functions whose distances to affine functions are all divisible by  $2^m$ . They deduced an upper bound on the number of  $m$ -resilient functions and improved upon Schneider's bound for  $m$  large.

In the present paper, we obtain an upper bound on  $m$ -resilient functions ( $m \geq \frac{n}{2} - 1$ ), and improve upon Schneider's bound for all values  $m > \frac{n}{2} - 1$ . We show with tables of values that our bound partially improves upon Carlet-Klapper's bound (the expressions of both bounds seem difficult to compare mathematically).

The organization of the paper is as follows. Section 2 introduces the notation and the definitions that are needed in the paper including the definition of correlation-immunity. Section 3 reviews the previous upper bounds on the numbers of first order correlation-immune functions, *i.e.*, Yang *et al*'s and Park *et al*'s bounds, and of  $m$ -resilient functions, *i.e.*, Schneider's and Carlet-Klapper's bounds. Extensions of Yang *et al*'s and Park *et al*'s bounds are given for the case of 1-resilient functions in this section for the first one and in appendix [B] for the second one. Section 4 introduces a new upper bound on the number of  $m$ -resilient functions. We give a table of values corresponding to the ratio of Schneider's bound to the new bound, and a second table corresponding to the ratio of Carlet-Klapper's bound to the new one.

## 2 Notation and Definitions

Let  $n$  be any positive integer. We denote by  $\oplus$  the usual addition in  $\mathbb{F}_2$  and in  $\mathbb{F}_2^n$ . The *Hamming weight*  $w_H(u)$  of a word  $u$  in  $\mathbb{F}_2^n$  is the number of its components equal to 1. We denote by  $\preceq$  the partial order on the words of  $\mathbb{F}_2^n$ , i.e.,  $(u_1, \dots, u_n) \preceq (v_1, \dots, v_n)$  if and only if  $(u_i = 1) \Rightarrow (v_i = 1)$ . Any Boolean function  $f$  in  $n$  variables,  $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ , admits a unique Algebraic Normal Form (A.N.F.):

$$f(x_1, \dots, x_n) = \bigoplus_{u \in \mathbb{F}_2^n} a_u \left( \prod_{i=1}^n x_i^{u_i} \right) = \bigoplus_{u \in \mathbb{F}_2^n} a_u x^u .$$

The function  $g : u \mapsto a_u$  is called the *Möbius transform* of  $f$ . For any word  $u$ , the coefficient  $a_u$  belongs to  $\mathbb{F}_2$ , and can be computed thanks to the formula

$$a_u = \bigoplus_{v \in \mathbb{F}_2^n, v \preceq u} f(v) . \quad (1)$$

The *algebraic degree* of a Boolean function  $f$  is the degree of its algebraic normal form. The *Hamming weight*  $w_H(f)$  of a Boolean function  $f$  in  $n$  variables is the size of its support, i.e., the size of the set  $\{x \in \mathbb{F}_2^n \mid f(x) = 1\}$ . A Boolean function  $f$  in  $n$  variables is called *balanced* if its Hamming weight equals  $2^{n-1}$ .

**Definition 1.** [11] Let  $X^{[j]} = (X_1^{[j]}, X_2^{[j]}, \dots, X_n^{[j]})$  be the  $n$ -tuple of LFSR output digits at time  $j$ . The combining function  $f$  is  $m$ th-order correlation-immune if every  $m$ -tuple obtained by fixing  $m$  components from  $X^{[j]}$  is statistically independent of the random value  $Z = f(X_1, X_2, \dots, X_n)$  associated to arbitrary outputs of LFSR.

A characterization of  $m$ th-order correlation-immune functions was given by Guo-Zhen and Massey in [4].

**Definition 2.** Let  $f$  be a Boolean function in  $n$  variables. The Walsh Transform of  $f$  is defined as the following real-valued function over the vector space  $\mathbb{F}_2^n$ ,

$$\hat{f}(x) = \sum_{u \in \mathbb{F}_2^n} f(u) (-1)^{u \cdot x} ,$$

where  $u \cdot x$  stands for  $\sum_{i=1}^n u_i x_i$ .

**Theorem 1.** [4] A Boolean combining function  $f$  in  $n$  variables is  $m$ th-order correlation-immune, where  $1 \leq m \leq n$ , if and only if for every word  $u$  in  $\mathbb{F}_2^n$  such that  $1 \leq w_H(u) \leq m$ ,  $\hat{f}(u)$  equals 0, i.e.,  $f(x) \oplus u \cdot x$  is balanced for all  $u$  such that  $1 \leq w_H(u) \leq m$ .

A balanced Boolean function in  $n$  variables which is correlation-immune of order  $m$  is called  $m$ -resilient. This notion was considered for the first time by Chor et al. in [2].

The tradeoff between the order of correlation-immunity and the algebraic degree was given by Siegenthaler.

**Theorem 2.** [12] *Let  $f$  be an  $m$ th-order correlation-immune Boolean function of degree  $d$  in  $n$  variables. Then  $d \leq n - m$ . Furthermore, if  $f$  is balanced then  $d \leq n - m - 1$  if  $m < n - 1$  and  $d = 1$  if  $m = n - 1$ .*

This result leads to the first obvious bound on the numbers of  $m$ -resilient and  $m$ th-order correlation-immune functions. The number of  $m$ th-order correlation-immune functions in  $n$  variables is upper bounded by  $2^{\sum_{i=0}^{n-m} \binom{n}{i}}$ , and the number of  $m$ -resilient functions in  $n$  variables is upper bounded by  $2^{\sum_{i=0}^{n-m-1} \binom{n}{i}}$  if  $m < n - 1$ .

### 3 Previous Upper Bounds

The number of  $m$ th-order correlation-immune Boolean functions is still unknown (an asymptotic formula is known, due to Denisov [3]). The first upper bound on the number of correlation-immune Boolean functions, published by Yang and Guo in 1990 [14], enumerates in fact the number of Boolean functions which satisfy partially the first order correlation-immunity criterion, *i.e.*, the functions  $f$  such that for two distinct integers  $i_1$  and  $i_2$ ,  $f \oplus x_{i_1}$  and  $f \oplus x_{i_2}$  are balanced. This leads to:

**Proposition 1.** [14] *Let  $n$  be a positive integer greater than 1. The number of 1st-order correlation-immune Boolean functions in  $n$  variables is less than:*

$$\sum_{k=0}^{2^{n-2}} \sum_{r=0}^k \binom{2^{n-2}}{r}^2 \binom{2^{n-2}}{k-r}^2.$$

Yang and Guo did not study the corresponding bound for 1-resilient functions. This can be done:

**Proposition 2.** *Let  $n$  be a positive integer greater than 1. The number of 1-resilient Boolean functions in  $n$  variables is less than:*

$$\sum_{a=0}^{2^{n-2}} \binom{2^{n-2}}{a}^4.$$

We give the proof of this bound in appendix A.

This work was deepened by Park, Lee, Sung and Kim for 1st-order correlation-immunity. They showed that the number of correlation-immune functions is itself upper bounded by this same number as in Proposition 2. Park *et al.* obtained this bound by numbering the Boolean functions such that for three distinct integers  $i_1, i_2, i_3$ , the functions  $f \oplus x_{i_1}$ ,  $f \oplus x_{i_2}$  and  $f \oplus x_{i_3}$  are balanced. They did not study the corresponding bound for 1-resilient Boolean functions. The bound obtained is quite complicated and is given in appendix B.

The number of balanced Boolean functions such that  $f(x) \oplus x_i$  is balanced for three distinct integers could be calculated by considering the solutions of a system of four equations with eight unknowns. When the number of integers

increases by one, only one new equation can be obtained and the number of unknowns is doubled. Thus, enumerating the number of Boolean functions such that  $f(x) \oplus x_i$  is balanced for  $i \in I \subseteq \{1, \dots, n\}$  leads to considering  $|I| + 1$  equations and  $2^{|I|}$  unknowns. Thus, for  $n$  greater than 3, the gap between the number of equations and the number of unknowns is too large to obtain a bound which can be computed easily.

Maitra and Sarkar [5] found a sufficient condition for a function  $f$  to be such that  $f(x) \oplus x_i$  is balanced for three values of  $i$  but  $f$  is not first order correlation-immune. A lower bound on the number of such functions provides an upper bound on the number of  $m$ th-order correlation-immune functions by using the bound of Park, Lee, Sung and Kim. However, the formula given by Maitra and Sarkar cannot be computed and thus their bound cannot be compared to the other bound.

Schneider proposed a new idea in 1990 for obtaining an upper bound on the number of  $m$ th-order correlation-immune Boolean functions, and an upper bound on  $m$ -resilient Boolean functions. In [10], he presented an algorithm for producing all correlation-immune functions. This algorithm is not very efficient (the workfactor, if computed, could be comparable to the complexity of searching among all Boolean functions). But the idea of this algorithm allowed him to provide an enumeration which is quite efficient.

**Theorem 3.** [10] *The number of  $m$ -resilient Boolean functions in  $n$  variables is less than:*

$$\prod_{i=1}^{n-m} \binom{2^i}{2^{i-1}}^{\binom{n-i-1}{m-1}}.$$

We can compare these three bounds by giving values in the 1-resilient case. It can be observed that Schneider’s bound is always better than Yang-Guo’s and Park *et al.*’s bounds for  $n > 4$ . The case  $n = 3$  can be explained: the number of balanced Boolean functions such that  $f(x) \oplus x_i$  is balanced for three distinct values of  $i$  is then exactly the number of 1-resilient functions.

Carlet and Klapper obtained two bounds on the number of  $m$ -resilient functions, one for  $2 \leq m < n/2$  and the other one for  $n/2 \leq m < n$ . They improved upon Schneider’s bound for  $m$  large.

**Table 1.** Values of previous upper bounds for first order resilient functions

$n$	YG (Resilient)	PLSK (Resilient)	Schneider
3	18	8	12
4	1810	648	840
5	$4.4916 \cdot 10^7$	$1.1979 \cdot 10^7$	$1.081 \cdot 10^7$
6	$7.0667 \cdot 10^{16}$	$1.3711 \cdot 10^{16}$	$6.498 \cdot 10^{15}$
7	$4.6909 \cdot 10^{35}$	$6.5259 \cdot 10^{34}$	$1.191 \cdot 10^{34}$
8	$5.6935 \cdot 10^{73}$	$5.6396 \cdot 10^{72}$	$2.8523 \cdot 10^{71}$

**Theorem 4.** □ *The number of  $m$ -resilient Boolean functions in  $n$  variables,  $n/2 \leq m < n$ , is less than:*

$$\frac{2^{1+\sum_{i=0}^{n-m-1} \binom{n}{i}} (1 + \varepsilon)}{2^{\sum_{i=0}^{n-m-1} \binom{m-1}{i}}} + 2^{\sum_{i=0}^{n-m-2} \binom{n}{i}},$$

where  $\varepsilon = \frac{1}{2^{\Omega((2^n/n)^{1/2})}}$ .

*The number of  $m$ -resilient Boolean functions in  $n$  variables,  $2 \leq m < n/2$ , is less than:*

$$\frac{2^{\sum_{i=0}^{n-m-1} \binom{n}{i}} - 2^{\sum_{i=0}^{n-m-2} \binom{n}{i}}}{2^{2^{m+1}-1}} + 2^{\sum_{i=0}^{n-m-2} \binom{n}{i}}.$$

## 4 A New Bound for $m$ -Resilient Functions

Our improvement of Schneider's bound is based on several ideas. One of them is to use more efficiently than Schneider does the bound on the degrees of  $m$ -resilient Boolean functions. Recall that, thanks to Siegenthaler's theorem, we know that, for  $m < n - 1$ , the degree of an  $m$ -resilient function in  $n$  variables is less than or equal to  $n - m - 1$ .

**Lemma 1.** *Let  $f$  be a Boolean function in  $n$  variables. If the algebraic degree of  $f$  is at most  $d$ , then  $f$  is completely determined by its values at the words  $u \in \mathbb{F}_2^n$  such that  $w_H(u) \leq d$ .*

*Proof.* Consider the algebraic normal form of the function:

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} g(u) x^u,$$

where  $g$  is the Möbius transform of  $f$ . For every word  $u$  such that  $d < w_H(u) \leq n$ , the coefficient  $g(u)$  is equal to zero, and thus:

$$\begin{aligned} f(x) &= \bigoplus_{u \in \mathbb{F}_2^n | w_H(u) \leq d} g(u) x^u \\ &= \bigoplus_{u \in \mathbb{F}_2^n | w_H(u) \leq d, u \preceq x} g(u) \\ &= \bigoplus_{u \in \mathbb{F}_2^n | w_H(u) \leq d, u \preceq x} \left( \bigoplus_{v \in \mathbb{F}_2^n | v \preceq u} f(v) \right). \end{aligned}$$

Every  $v$  such that  $v \preceq u$  where  $w_H(u) \leq d$  has weight at most  $d$ . □

The number of Boolean functions of degrees less than  $n - m - 1$  being negligible in comparison with Schneider's bound, we shall bound the number of  $m$ -resilient functions of degree exactly  $n - m - 1$  and add the number of Boolean functions of degrees less than  $n - m - 1$ . To this aim, we shall use a lemma which was first proved in □. But we shall need a slightly different statement of this lemma, with extra precisions that will be useful in our context. For this reason, we give a proof of the lemma. We first introduce a notation:

Let  $u$  and  $v$  be two vectors in  $\mathbb{F}_2^n$ ; we denote by  $v \wedge u$  the vector such that, for every index  $i$ ,  $(v \wedge u)_i = v_i u_i = \min(v_i, u_i)$ , i.e., and by  $v \vee u$  the vector such that, for every index  $i$ ,  $(v \vee u)_i = \max(v_i, u_i)$  (these two operations are called bitwise-AND and bitwise-OR).

**Lemma 2.** [1] Let  $f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u x^u$  be an  $m$ -resilient Boolean function in  $n$  variables of degree  $n - m - 1 \geq 2$  with  $m \geq 2$  and let  $\bigoplus_{u \in \mathbb{F}_2^n} b_u x^u$  be the ANF of the function  $f(x) \oplus x_1 \oplus \cdots \oplus x_n$  (i.e.  $b_u = a_u$  if  $w_H(u) > 1$  or if  $u = 0$  and  $b_u = a_u \oplus 1$  if  $w_H(u) = 1$ ). If  $u$  is a word in  $\mathbb{F}_2^n$  of weight  $n - m - 1$  such that  $a_u = 1$  (i.e.  $b_u = 1$ ) then for all non-zero  $v$  in  $\mathbb{F}_2^n$  such that  $v \wedge u = 0$ , we have:

$$b_v = \bigoplus_{\substack{s \vee t = u \vee v \\ s \wedge u \neq 0 \\ t \wedge u \neq 0}} b_s b_t .$$

*Proof.* We know (cf. [1]) that for every word  $x$  such that  $w_H(x) \geq n - m$ , we have that  $\bigoplus_{\{s, t\} | s \vee t = x} b_s b_t = 0$ . We apply this to  $x = v \vee u$ . In the corresponding relation, the coefficient  $b_v$  appears with a non-zero coefficient only in the term  $b_u b_v$  since if  $b_{u'}$   $b_v$  appears, then  $u' \vee v = x$ , so  $u \preceq u'$ . We deduce:

$$b_v = \bigoplus_{\substack{s \vee t = u \vee v \\ s, t \neq v}} b_s b_t .$$

According to Siegenthaler's inequality, the double condition that  $s \vee t = u \vee v$  and  $s, t \neq v$  implies, if  $b_s \neq 0$  and  $b_t \neq 0$ , that  $s \wedge u \neq 0$  and  $t \wedge u \neq 0$  since  $u$  has weight  $n - m - 1$ .  $\square$

**Theorem 5.** Let  $n$  and  $m$  be two positive integers such that  $\frac{n}{2} - 1 \leq m < n - 2$ . The number of  $m$ -resilient functions of degree  $n - m - 1$  in  $n$  variables is lower than:

$$\frac{\binom{n}{n-m-1}}{2^{\binom{m+1}{n-m-1}+1}} \prod_{i=1}^{n-m} \binom{2^i}{2^{i-1}}^{\binom{n-i-1}{m-1}} .$$

Thus, the number of  $m$ -resilient functions in  $n$  variables is lower than:

$$2^{\sum_{i=0}^{n-m-2} \binom{n}{i}} + \frac{\binom{n}{n-m-1}}{2^{\binom{m+1}{n-m-1}+1}} \prod_{i=1}^{n-m} \binom{2^i}{2^{i-1}}^{\binom{n-i-1}{m-1}} .$$

The principle of the proof is to bound the number of different truth-tables of  $m$ -resilient functions of maximum degree ( $d = n - m - 1$ ) by using the fact that some of their successive restrictions are balanced. The bound is then obtained by adding the number of Boolean functions of degrees at most  $n - m - 2$  (which is negligible).

*Proof.* According to Siegenthaler's Theorem on the degrees of resilient functions and according to Lemma 1, we only need, when evaluating the number of possible truth tables of  $f$  (that is the number of choices of the values of  $f$  at words  $u \in \mathbb{F}_2^n$ ) to consider the words  $u$  such that  $0 \leq w_H(u) \leq n - m - 1$ . In order to bound the number of  $m$ -resilient functions of degree exactly  $n - m - 1$ , we first bound the number of  $m$ -resilient functions whose ANF contains the monomial  $x_1 \dots x_{n-m-1}$ . We proceed by induction.

- Step 1: Every  $m$ -resilient Boolean function  $f$  is such that the restricted function  $f(x_1, \dots, x_{n-m}, 0, \dots, 0)$  is balanced, *i.e.*, has weight  $2^{n-m-1}$ . Since the monomial  $x_1 \dots x_{n-m-1}$  appears in the ANF of the function, the number of words of the support which are less than  $u = 1^{n-m-1}0^{m+1}$  for the partial order is odd. Consequently, there are

$$\sum_{i \text{ odd}} \binom{2^{n-m-1}}{i} \binom{2^{n-m-1}}{2^{n-m-1}-i} = \frac{1}{2} \binom{2^{n-m}}{2^{n-m-1}}$$

different choices for the restriction of the truth-table of  $f$  at words of  $\{0, 1\}^{n-m} \times \{0\}^m$ .

- Step 2: We now consider the restrictions of  $f$  in which the  $(n-m)$ th variable is fixed to zero. For the values of the variables  $x_{n-m+1}, \dots, x_n$ , we fix  $m-1$  variables among  $m$  to zero (there are  $m$  possible different choices), and the last free one is fixed to 1 because the cases where it is fixed to 0 have already been considered at the previous step. Indeed every word  $v$  lower (for the partial order  $\preceq$ ) than the word  $u = 1^{n-m}0^m$  has been considered at the first step and, *a fortiori*, every word lower than  $u' = 1^{n-m-1}0^{m+1}$  has already been considered.

Thus only the words in  $\{u \in \mathbb{F}_2^n | u = (u_1, \dots, u_{n-m-1}, 0, \dots, 0, 1, 0, \dots, 0)\}$  will be given a value by  $f$  at this step. We do not know how many words in this set must be in the support of the considered functions since we do not know how many words in the set  $\{u \in \mathbb{F}_2^n | u = (u_1, \dots, u_{n-m-1}, 0, \dots, 0)\}$  are already in the support. But if this latter number is  $i$ , then the former one must be  $j = 2^{n-m-1} - i$ . And we know that for every  $j$  we have:

$$\binom{2^{n-m-1}}{j} \leq \binom{2^{n-m-1}}{2^{n-m-2}}.$$

We can bound the number of choices for one such restriction by  $\binom{2^{n-m-1}}{2^{n-m-2}}$ , and since the number of such restrictions is  $m$ , the number of choices  $\binom{2^{n-m-1}}{2^{n-m-2}}$  is raised to the  $m$ th power. At the end of this step, we have considered all the words in  $\mathbb{F}_2^n$  such that  $0 \leq w_H(x_{n-m}, x_{n-m+1}, \dots, x_n) \leq 1$ .

- Step  $p$ : Assume we have already chosen the values on the words  $x$  such that  $0 \leq w_H(x_{n-m-p+3}, \dots, x_n) \leq p-2$ .

We now consider the restrictions such that  $x_{n-m-p+2} = 0$ , and  $m-1$  variables among  $m+p-2$  are fixed to 0; the remaining free variables are fixed to 1 because the other cases have already been considered in the previous steps. Thus there are  $\binom{m+p-2}{m-1}$  such restrictions. For each restriction, we do not know exactly how many words should be in the support, but this number can be bounded by the maximum possible number of choices, *i.e.*,  $\binom{2^{n-m-p+1}}{2^{n-m-p}}$ . Since there are  $\binom{m+p-2}{m-1}$  such restrictions, the number of choices  $\binom{2^{n-m-p+1}}{2^{n-m-p}}$  is raised to the power  $\binom{m+p-2}{m-1}$ . We show now that, at the end of this step, we have considered all the words  $x$  such that  $0 \leq w_H(x_{n-m-p+2}, \dots, x_n) \leq p-1$ : if  $w_H(x_{n-m-p+2}, \dots, x_n) \leq p-2$  or if  $w_H(x_{n-m-p+2}, \dots, x_n) = p-1$  and  $x_{n-m-p+2} = 1$ , then  $x$  has been considered before step  $p$  (by induction hypothesis); and if  $w_H(x_{n-m-p+2}, \dots, x_n) = p-1$  and  $x_{n-m-p+2} = 0$ , then it has been considered at step  $p$ .

- Step  $n - m$ : According to the property proved above, all the words such that  $w_H(x_3, \dots, x_n) \leq n - m - 2$  have been considered at the end of step  $n - m - 1$ . Thus, only the words of weight  $n - m - 1$  and such that  $x_1 = x_2 = 0$  have still to be given a value by  $f$ . We first choose a value  $f(x)$  for every word  $x = (0, 0, x_3, \dots, x_n)$  of weight  $n - m - 1$  such that  $x \wedge u \neq 0$ , where  $u = 1^{n-m-1}0^{m+1}$ . The number of such choices equals  $2^{\binom{n-2}{n-m-1} - \binom{m+1}{n-m-1}}$ . We apply now Lemma 2 to any word  $v$  of weight  $n - m - 1$  and such that  $v \wedge u = 0$ . We deduce the value of  $b_v$  and thus of  $a_v$ . Indeed, according to relation (II), the values of all the bits  $b_s, b_t$  such that  $s \vee t = u \vee v$  and  $s \wedge u \neq 0, t \wedge u \neq 0$  can be deduced from the values of  $f(x)$  already chosen since  $x \preceq s$  implies that either  $w_H(x) < n - m - 1$  or  $x \wedge u \neq 0$ . The knowledge of  $b_v$  implies that of  $f(v)$  because all the values  $f(x)$  such that  $x \prec v$  have been already chosen, and according to relation (II).

We have now proved that the number of  $m$ -resilient functions  $f$  of degree  $n - m - 1$  and whose ANF contains the monomial  $x_1 \dots x_{n-m-1}$  is upper bounded by:

$$\frac{1}{2^{\binom{m+1}{n-m-1}+1}} \prod_{i=1}^{n-m} \binom{2^i}{2^{i-1}}^{\binom{n-i-1}{m-1}}.$$

This number does not change if we replace the monomial  $x_1 \dots x_{n-m-1}$  by any other monomial  $\mu$  of same degree (since the notion of resiliency is invariant under the permutation of the coordinates of  $x$ ). Any  $m$ -resilient function of degree  $n - m - 1$  belonging to  $\bigcup_{\mu} S_{\mu}$ , where  $S_{\mu}$  is the set of all  $m$ -resilient functions of degree  $n - m - 1$  whose ANF contains  $\mu$ , we obtain a bound on the number of  $m$ -resilient functions of degree  $n - m - 1$  by multiplying the number above by the number of these monomials, *i.e.*,  $\binom{n}{n-m-1}$ . Our bound on the number of all  $m$ -resilient functions is then obtained by adding the number of Boolean functions of degrees at most  $n - m - 2$ .  $\square$

We now give tables of values permitting to compare the bounds. We give in the first table the values of the new bound for  $\lceil \frac{n}{2} \rceil \leq m \leq \lceil \frac{n}{2} \rceil + 5$ . In the next table, we compare Schneider's bound and the new bound (which improves upon it for  $m \geq \lceil \frac{n}{2} \rceil$ ). In the last table of values, we compare Carlet-Klapper's bound and the new one.

*Remark 1.* A slight improvement of our bound is possible: let  $k$  be a positive integer; the number of Boolean functions of degree at most  $n - m - 1$  and whose ANF contains at most  $k - 1$  monomials of degree  $n - m - 1$  equals  $2^{\sum_{i=0}^{n-m-2} \binom{n}{i} \left( \sum_{j=0}^{k-1} \binom{n-m-1}{j} \right)}$ . We deduce that the number of  $m$ -resilient functions in  $n$  variables is lower than:

$$2^{\sum_{i=0}^{n-m-2} \binom{n}{i} \left( \sum_{j=0}^{k-1} \binom{n-m-1}{j} \right)} + \frac{\binom{n}{n-m-1}}{k 2^{\binom{m+1}{n-m-1}+1}} \prod_{i=1}^{n-m} \binom{2^i}{2^{i-1}}^{\binom{n-i-1}{m-1}}.$$

We have checked that for almost every  $n$ , some values of  $k \leq \binom{n}{n-m-1}$  permit to improve upon our bound.



**Table 2.** New bound on the number of  $m$ -resilient functions

$n \setminus m$	$\lceil \frac{n}{2} \rceil$	$\lceil \frac{n}{2} \rceil + 1$	$\lceil \frac{n}{2} \rceil + 2$	$\lceil \frac{n}{2} \rceil + 3$	$\lceil \frac{n}{2} \rceil + 4$	$\lceil \frac{n}{2} \rceil + 5$
6	$1.1 \cdot 10^5$	11	—	—	—	—
7	$9.5 \cdot 10^5$	12	—	—	—	—
8	$5.36 \cdot 10^{23}$	$7.6 \cdot 10^6$	14	—	—	—
9	$1.4 \cdot 10^{31}$	$5.9 \cdot 10^7$	15	—	—	—
10	$6.5 \cdot 10^{102}$	$4.2 \cdot 10^{39}$	$4.4 \cdot 10^8$	17	—	—
11	$2.3 \cdot 10^{145}$	$1.4 \cdot 10^{49}$	$3.2 \cdot 10^9$	18	—	—
12	$5.6 \cdot 10^{430}$	$1.3 \cdot 10^{199}$	$5.8 \cdot 10^{59}$	$2.3 \cdot 10^{10}$	20	—
13	$1.6 \cdot 10^{638}$	$2.6 \cdot 10^{265}$	$2.7 \cdot 10^{71}$	$1.6 \cdot 10^{11}$	21	—
14	$1.3 \cdot 10^{1776}$	$1.3 \cdot 10^{918}$	$4.8 \cdot 10^{345}$	$1.5 \cdot 10^{84}$	$1.2 \cdot 10^{12}$	23
15	$3.4 \cdot 10^{2712}$	$3.7 \cdot 10^{1286}$	$1.9 \cdot 10^{441}$	$9.7 \cdot 10^{97}$	$8.0 \cdot 10^{12}$	47
16	$3.8 \cdot 10^{7264}$	$1.2 \cdot 10^{4034}$	$2.0 \cdot 10^{1761}$	$3.9 \cdot 10^{553}$	$7.5 \cdot 10^{112}$	$5.5 \cdot 10^{13}$
17	$1.6 \cdot 10^{11333}$	$2.7 \cdot 10^{5855}$	$5.7 \cdot 10^{2361}$	$1.0 \cdot 10^{684}$	$6.8 \cdot 10^{128}$	$3.7 \cdot 10^{14}$
18	$7.6 \cdot 10^{29577}$	$8.2 \cdot 10^{17260}$	$1.6 \cdot 10^{8313}$	$1.1 \cdot 10^{3109}$	$7.8 \cdot 10^{833}$	$7.3 \cdot 10^{145}$
19	$1.1 \cdot 10^{46898}$	$2.8 \cdot 10^{25709}$	$6.5 \cdot 10^{11567}$	$9.8 \cdot 10^{4025}$	$4.4 \cdot 10^{1004}$	$9.4 \cdot 10^{163}$
20	$7.8 \cdot 10^{120074}$	$2.4 \cdot 10^{72742}$	$5.3 \cdot 10^{37511}$	$2.7 \cdot 10^{15805}$	$1.1 \cdot 10^{5137}$	$4.3 \cdot 10^{1197}$
21	$1.2 \cdot 10^{192912}$	$7.7 \cdot 10^{110527}$	$3.0 \cdot 10^{53700}$	$1.2 \cdot 10^{21240}$	$2.5 \cdot 10^{6468}$	$1.8 \cdot 10^{1414}$

**Table 3.** (Schneider's bound/new bound) for  $m$ -resilient functions

$n \setminus m$	$\lceil \frac{n}{2} \rceil$	$\lceil \frac{n}{2} \rceil + 1$	$\lceil \frac{n}{2} \rceil + 2$	$\lceil \frac{n}{2} \rceil + 3$	$\lceil \frac{n}{2} \rceil + 4$	$\lceil \frac{n}{2} \rceil + 5$
6	8.5	8.7	—	—	—	—
7	$9.8 \cdot 10^1$	$1.5 \cdot 10^1$	—	—	—	—
8	$3.7 \cdot 10^1$	$2.3 \cdot 10^3$	$2.7 \cdot 10^1$	—	—	—
9	$2.5 \cdot 10^4$	$1.2 \cdot 10^5$	$5.0 \cdot 10^1$	—	—	—
10	$3.1 \cdot 10^2$	$5.7 \cdot 10^8$	$1.2 \cdot 10^7$	$9.0 \cdot 10^1$	—	—
11	$2.1 \cdot 10^8$	$8.7 \cdot 10^{14}$	$2.5 \cdot 10^9$	$1.7 \cdot 10^2$	—	—
12	$5.3 \cdot 10^3$	$4.8 \cdot 10^{18}$	$1.8 \cdot 10^{23}$	$1.1 \cdot 10^{12}$	$3.1 \cdot 10^2$	—
13	$1.1 \cdot 10^{14}$	$2.4 \cdot 10^{35}$	$9.3 \cdot 10^{33}$	$9.2 \cdot 10^{14}$	$5.7 \cdot 10^2$	—
14	$1.8 \cdot 10^5$	$8.5 \cdot 10^{34}$	$3.3 \cdot 10^{60}$	$2.6 \cdot 10^{47}$	$1.6 \cdot 10^{18}$	$1.1 \cdot 10^3$
15	$7.7 \cdot 10^{21}$	$4.8 \cdot 10^{72}$	$3.2 \cdot 10^{96}$	$7.4 \cdot 10^{63}$	$5.7 \cdot 10^{21}$	$2.0 \cdot 10^3$
16	$1.2 \cdot 10^7$	$4.1 \cdot 10^{59}$	$5.4 \cdot 10^{135}$	$1.1 \cdot 10^{146}$	$4.4 \cdot 10^{83}$	$4.1 \cdot 10^{25}$
17	$1.3 \cdot 10^{32}$	$1.9 \cdot 10^{135}$	$8.4 \cdot 10^{234}$	$1.4 \cdot 10^{212}$	$1.1 \cdot 10^{107}$	$6.0 \cdot 10^{29}$
18	$1.6 \cdot 10^9$	$1.4 \cdot 10^{95}$	$1.5 \cdot 10^{274}$	$6.2 \cdot 10^{383}$	$1.4 \cdot 10^{298}$	$2.3 \cdot 10^{134}$
19	$1.2 \cdot 10^{45}$	$1.0 \cdot 10^{234}$	$2.7 \cdot 10^{512}$	$7.9 \cdot 10^{598}$	$4.2 \cdot 10^{407}$	$7.8 \cdot 10^{165}$
20	$4.3 \cdot 10^{11}$	$1.6 \cdot 10^{144}$	$9.5 \cdot 10^{511}$	$5.1 \cdot 10^{899}$	$1.3 \cdot 10^{900}$	$3.1 \cdot 10^{544}$
21	$1.1 \cdot 10^{61}$	$2.6 \cdot 10^{382}$	$2.3 \cdot 10^{1028}$	$1.7 \cdot 10^{1503}$	$7.8 \cdot 10^{1310}$	$9.4 \cdot 10^{712}$

## 5 Conclusion

We have obtained for  $m \geq \frac{n}{2}$  an improvement of Schneider's bound on the number of  $m$ -resilient functions in  $n$  variables. The tables computed show that our bound also partially improves upon Carlet-Klapper's bound. Notice that the values of  $m$  for which this happens in the tables are those among which the best satisfactory tradeoffs between resiliency order, nonlinearity (limited by Sarkar-Maitra's bound [9]) and degree (limited by Siegenthaler's bound) can

**Table 4.** (Carlet-Klapper's bound/new bound) for  $m$ -resilient functions

$n \backslash m$	$\lceil \frac{n}{2} \rceil$	$\lceil \frac{n}{2} \rceil + 1$	$\lceil \frac{n}{2} \rceil + 2$	$\lceil \frac{n}{2} \rceil + 3$	$\lceil \frac{n}{2} \rceil + 4$	$\lceil \frac{n}{2} \rceil + 5$
6	$5.5 \cdot 10^{-2}$	1.27	—	—	—	—
7	$2.6 \cdot 10^{-2}$	1.12	—	—	—	—
8	$5.5 \cdot 10^{-4}$	$1.2 \cdot 10^{-2}$	1	—	—	—
9	$4.4 \cdot 10^{-5}$	$6.7 \cdot 10^{-3}$	$9.0 \cdot 10^{-1}$	—	—	—
10	$3.3 \cdot 10^{-4}$	$2.4 \cdot 10^{-6}$	$3.6 \cdot 10^{-3}$	$8.2 \cdot 10^{-1}$	—	—
11	$2.0 \cdot 10^{-6}$	$9.7 \cdot 10^{-8}$	$1.9 \cdot 10^{-3}$	$7.6 \cdot 10^{-1}$	—	—
12	$7.3 \cdot 10^{10}$	$1.4 \cdot 10^{-9}$	$2.6 \cdot 10^{-9}$	$1.1 \cdot 10^{-3}$	$7.0 \cdot 10^{-1}$	—
13	$4.1 \cdot 10^{12}$	$7.0 \cdot 10^{-14}$	$4.7 \cdot 10^{-11}$	$6.1 \cdot 10^{-4}$	$6.5 \cdot 10^{-1}$	—
14	$2.0 \cdot 10^{99}$	$4.5 \cdot 10^{12}$	$1.7 \cdot 10^{-19}$	$5.9 \cdot 10^{-13}$	$3.5 \cdot 10^{-4}$	$6.1 \cdot 10^{-1}$
15	$2.7 \cdot 10^{142}$	$9.1 \cdot 10^9$	$1.3 \cdot 10^{-26}$	$5.0 \cdot 10^{-15}$	$2.0 \cdot 10^{-4}$	$5.7 \cdot 10^{-1}$
16	$4.2 \cdot 10^{511}$	$1.6 \cdot 10^{194}$	$2.2 \cdot 10^3$	$2.4 \cdot 10^{-35}$	$2.9 \cdot 10^{-17}$	$1.2 \cdot 10^{-4}$
17	$9.3 \cdot 10^{785}$	$2.3 \cdot 10^{253}$	$2.5 \cdot 10^{-9}$	$6.1 \cdot 10^{-46}$	$1.1 \cdot 10^{-19}$	$6.9 \cdot 10^{-5}$
18	$1.1 \cdot 10^{2256}$	$6.4 \cdot 10^{1158}$	$4.1 \cdot 10^{317}$	$3.9 \cdot 10^{-28}$	$1.4 \cdot 10^{-58}$	$2.9 \cdot 10^{-22}$
19	$3.7 \cdot 10^{3610}$	$1.2 \cdot 10^{1649}$	$2.4 \cdot 10^{383}$	$1.6 \cdot 10^{-55}$	$2.2 \cdot 10^{-73}$	$5.1 \cdot 10^{-25}$
20	$2.3 \cdot 10^{9330}$	$5.9 \cdot 10^{5571}$	$1.3 \cdot 10^{2275}$	$4.7 \cdot 10^{445}$	$2.0 \cdot 10^{-93}$	$1.5 \cdot 10^{-90}$
21	$5.4 \cdot 10^{15353}$	$2.5 \cdot 10^{8328}$	$4.1 \cdot 10^{3053}$	$8.1 \cdot 10^{497}$	$7.1 \cdot 10^{-144}$	$2.6 \cdot 10^{-110}$
22	$3.7 \cdot 10^{37456}$	$1.5 \cdot 10^{24442}$	$5.2 \cdot 10^{12102}$	$2.0 \cdot 10^{3998}$	$5.2 \cdot 10^{532}$	$3.6 \cdot 10^{-209}$

be obtained (since none of these parameters must be small). Moreover, we can conjecture that, asymptotically, the new bound improves upon Carlet-Klapper's bound when  $m - n/2$  is fixed and  $n$  tends to infinity (recall that Carlet-Klapper's bound improves upon Schneider's one when  $n - m$  is fixed and  $n$  tends to infinity).

## References

1. Carlet, C., Klapper, A.: Upper bounds on the number of resilient functions and of bent functions. Proceedings of 23rd Symposium on Information Theory in the Benelux (2002)
2. Chor, B., Goldreich, O., Hastad, J., Rudich, S., Smolensky, R.: The bit extraction problem or  $t$ -resilient functions. In 26th IEEE Symposium on Foundations of Computer Science (1985) 396–407
3. Denisov, O.: An asymptotic formula for the number of correlation-immune of order  $k$  Boolean functions. Discrete Mathematics and Applications **2** (4) (1992) 407–426
4. Guo-Zhen, X., Massey, J.L.: A spectral characterization of correlation-immune combining functions. IEEE Transaction on Information Theory **34** (3) (1988) 569–571
5. Maitra, S., Sarkar, P.: Enumeration of correlation-immune Boolean functions. ACISP (1999) 12–25
6. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Transactions on Information Theory **15** (1969) 122–127
7. Mitchell, C.J.: Enumerating Boolean functions of cryptographic significance. Journal of Cryptology **2** (3) (1990) 155–170

8. Park, S. M., Lee, S., Sung, S. H., Kim, K.: Improving bounds for the number of correlation-immune Boolean functions. *Information Processing Letters* **61** (1997) 209–212
9. Sarkar, P., Maitra, S.: Nonlinearity bounds and constructions of resilient Boolean functions. *Crypto 2000* (2000) 515–532
10. Schneider, M.: A note on the construction and upper bounds of correlation-immune functions. 6th IMA Conference (1997) 295–306
11. Siegenthaler, T.: Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory* **30** (5) (1984) 776–780
12. Siegenthaler, T.: Decrypting a class of stream cipher using ciphertext only. *IEEE Transactions on Computers* **34** (1) (1985) 81–85
13. Vernam, G.: Cipher printing telegraph systems for secret wire and radio telegraphic communication. *Journal of the American Institute of Electrical Engineers* **45** (1926) 109–115
14. Yang, Y. X., Guo, B.: Further enumerating Boolean functions of cryptographic significance. *Journal of Cryptology* **8** (3) (1995) 115–122

## A Proof of Proposition 2

We prove that the number of 1-resilient functions in  $n$  variables is less than  $\sum_{k=0}^{2^{n-2}} \binom{2^{n-2}}{k}^4$ .

Every Boolean function  $f$  in  $n$  variables can be considered as the concatenation of four Boolean functions in  $n-2$  variables,  $f = f_1 f_2 f_3 f_4$ . The ANF of the function is

$$f = (1 - x_n)(1 - x_{n-1})f_1 \oplus (1 - x_n)x_{n-1}f_2 \oplus x_n(1 - x_{n-1})f_3 \oplus x_n x_{n-1}f_4 .$$

We have:

$$w_H(f) = 2^{n-1} \Leftrightarrow w_H(f_1) + w_H(f_2) + w_H(f_3) + w_H(f_4) = 2^{n-1} \quad (2)$$

$$w_H(f|_{x_n=0}) = 2^{n-2} \Leftrightarrow w_H(f_1) + w_H(f_2) = 2^{n-2} \quad (3)$$

$$w_H(f|_{x_{n-1}=0}) = 2^{n-2} \Leftrightarrow w_H(f_1) + w_H(f_3) = 2^{n-2} \quad (4)$$

Thus,

$$(3), (4) \Rightarrow w_H(f_2) = w_H(f_3) \quad (5)$$

$$(2), (3), (5) \Rightarrow w_H(f_1) = w_H(f_4) \quad (6)$$

The bound of Proposition 2 is then a direct consequence of equations (3), (5) and (6). Indeed, we can deduce:

$$\sum_{w_H(f_1)=0}^{2^{n-2}} \binom{2^{n-2}}{w_H(f_1)}^2 \binom{2^{n-2}}{2^{n-2} - w_H(f_1)}^2 .$$

□

## B Park, Lee Sung and Kim's Bound in the Case of First Order Resilient Boolean Function

**Proposition 3.** *Let  $n$  be a positive integer greater than 1. The number of 1-resilient Boolean functions in  $n$  variables is less than:*

$$\sum_{a,b,c,d=0}^{2^{n-3}} \binom{2^{n-3}}{a} \binom{2^{n-3}}{b} \binom{2^{n-3}}{c} \binom{2^{n-3}}{d} \binom{2^{n-3}}{2^{n-2}-a-b-c} \\ \times \binom{2^{n-3}}{2^{n-2}-a-c-d} \binom{2^{n-3}}{c+d-b} \binom{2^{n-3}}{a+b-d}.$$

*Proof.* Every Boolean function in  $n$  variables  $f$  can be considered as the concatenation of eight functions in  $n-3$  variables, i.e.,  $f = f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8$ . The corresponding ANF of the function is

$$\begin{aligned} f = & (1-x_n)(1-x_{n-1})(1-x_{n-2})f_1 \oplus (1-x_n)(1-x_{n-1})x_{n-2}f_2 \\ & \oplus (1-x_n)x_{n-1}(1-x_{n-2})f_3 \oplus (1-x_n)x_{n-1}x_{n-2}f_4 \\ & \oplus x_n(1-x_{n-1})(1-x_{n-2})f_5 \oplus x_n(1-x_{n-1})x_{n-2}f_6 \\ & \oplus x_nx_{n-1}(1-x_{n-2})f_7 \oplus x_nx_{n-1}x_{n-2}f_8. \end{aligned}$$

We have the following equations:

$$w_H(f) = 2^{n-1} \Leftrightarrow \sum_{i=1}^8 w_H(f_i) = 2^{n-1} \quad (7)$$

$$w_H(f|_{x_n=0}) = 2^{n-2} \Leftrightarrow w_H(f_1) + w_H(f_2) + w_H(f_3) + w_H(f_4) = 2^{n-2} \quad (8)$$

$$w_H(f|_{x_{n-1}=0}) = 2^{n-2} \Leftrightarrow w_H(f_1) + w_H(f_2) + w_H(f_5) + w_H(f_6) = 2^{n-2} \quad (9)$$

$$w_H(f|_{x_{n-2}=0}) = 2^{n-2} \Leftrightarrow w_H(f_1) + w_H(f_3) + w_H(f_5) + w_H(f_7) = 2^{n-2} \quad (10)$$

We obtain:

$$(\text{8}), (\text{9}) \Rightarrow w_H(f_3) + w_H(f_4) = w_H(f_5) + w_H(f_6) \quad (11)$$

$$(\text{7}), (\text{11}) \Rightarrow w_H(f_1) + w_H(f_2) = w_H(f_7) + w_H(f_8) \quad (12)$$

Assume that the values of  $w_H(f_1)$ ,  $w_H(f_2)$ ,  $w_H(f_3)$  and  $w_H(f_7)$  are fixed, then

$$(\text{8}) \Rightarrow w_H(f_4) = 2^{n-2} - w_H(f_1) - w_H(f_2) - w_H(f_3) \quad (13)$$

$$(\text{10}) \Rightarrow w_H(f_5) = 2^{n-2} - w_H(f_1) - w_H(f_3) - w_H(f_7) \quad (14)$$

$$(\text{9}), (\text{14}) \Rightarrow w_H(f_6) = w_H(f_3) + w_H(f_7) - w_H(f_2) \quad (15)$$

$$(\text{12}) \Rightarrow w_H(f_8) = w_H(f_1) + w_H(f_2) - w_H(f_7) \quad (16)$$

Since we know that the values of  $w_H(f_1)$ ,  $w_H(f_2)$ ,  $w_H(f_3)$  and  $w_H(f_7)$  vary between 0 and  $2^{n-3}$ , we can deduce the formula with the equations (13), (14), (15) and (16).  $\square$

# Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks

Emmanuel Bresson<sup>1</sup>, Olivier Chevassut<sup>2</sup>, and David Pointcheval<sup>1</sup>

<sup>1</sup> École normale supérieure, 75230 Paris Cedex 05, France,  
{Emmanuel.Bresson,David.Pointcheval}@ens.fr,  
<http://www.di.ens.fr/~{bresson,pointcheval}>

<sup>2</sup> Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA,  
OChevassut@lbl.gov,  
<http://www.itg.lbl.gov/~chevassu>

**Abstract.** Group Diffie-Hellman schemes for password-based key exchange are designed to provide a pool of players communicating over a public network, and sharing just a human-memorable password, with a session key (e.g. the key is used for multicast data integrity and confidentiality). The fundamental security goal to achieve in this scenario is security against dictionary attacks. While solutions have been proposed to solve this problem no formal treatment has ever been suggested. In this paper, we define a security model and then present a protocol with its security proof in both the random oracle model and the ideal-cipher model.

## 1 Introduction

Group Diffie-Hellman schemes for password-based key exchange are designed to provide a pool of players, communicating over a public network, and holding a shared human-memorable password with a session key to be used to implement secure multicast sessions. A human-memorable password  $pw$  is a (short) string chosen from a relatively small dictionary to be easily memorized and typed-in by a human.

Consider mission-critical applications such as emergency rescue and military operations [18,19,21], or even commercial applications like conferencing/meeting [11,19] and personal networking [5,13], where a (small) group of people collaborate. These applications operate in a highly mobile environment characterized by the lack of any fixed network and security infrastructure. At the same time, these are applications where secure multicast sessions may be needed. Due to the absence of fixed infrastructure, session keys can be computed via a group Diffie-Hellman key exchange bootstrapped from a password. A password usually chosen by the participants may be a low-quality one (i.e. 4 decimal digits) easier to memorize than a high-quality one (i.e. 56-bit, 192-bit).

The fundamental security goal for a group Diffie-Hellman protocol designed for such a scenario to achieve is security against dictionary attacks. One can not actually prevent the adversary from guessing a value for  $pw$  and using this

value in an attempt to impersonate a player. If the attack fails, the adversary can eliminate this value from the list of possible values for  $pw$ . However, one would like this attack to be the only one the adversary can mount: after  $n$  active interactions with some participants the adversary should not be able to eliminate a greater number of passwords than  $n$ . Namely, a passive eavesdropping should be of no help to the adversary since an off-line exhaustive search on  $pw$  should not get any bias on the actual password – such a bias could be later used in on-line interactions. The off-line exhaustive search is called *dictionary attack*.

**Contributions.** This paper represents the first formal treatment of the authenticated group Diffie-Hellman key exchange problem when the parties share a human-memorable password. We start from the model of Bresson et al. [10] and enhance it to capture dictionary attacks. In our model, the parties are modeled through oracles and the various types of attacks are modeled by queries to these oracles. The model is equipped with the ability to obtain honest protocol executions to enable a treatment of dictionary attacks.

Our model is used to define the execution of a password-based group Diffie-Hellman protocol which we refer to as EKE (*Encrypted Key Exchange*, see [3]). Converting a provably authenticated group Diffie-Hellman protocol [10] into a password-based group Diffie-Hellman protocol is not an easy task. The trivial conversion consisting in substituting a signature scheme by a symmetric encryption scheme, using the password as secret key as for the two-party case [27], does not provide security against dictionary attacks. We have, in effect, to perform several modifications to the protocol of Bresson et al. [10]. The modifications cost only one more exponentiation per player however we also notice that the cost of the signatures and verifications is replaced by the cost of a symmetric encryption, which is very low. The flows are moreover shorter since there is no longer a signature.

The security against dictionary attacks shows up in Theorem 1 which asserts the security of EKE in both the random oracle model and the ideal-cipher model. Security against dictionary attacks depends on how many interactions the adversary carries out against the instances rather than on the adversary’s computational power. The theorem exhibits a reduction from the semantic security of an EKE session key to reasonable and well-defined computational problems.

Our paper is organized as follows. In the remainder of this section we summarize the related work. In Section 2, we define our model and the definitions that should be satisfied by a group Diffie-Hellman scheme secure against dictionary attacks. In Section 3, we present the intractability assumptions we use in this paper. We present the EKE protocol in Section 4 and assert its security in both the random oracle model and the ideal-cipher model in Section 5. We then prove its security in Section 6. Finally, some extensions are provided: we briefly deal with forward-secrecy in Section 7 and with mutual authentication in Section 8.

**Related Work.** Several 2-party Diffie-Hellman key exchange protocols aimed to distribute a session key among two parties when the parties share a password. Recently, Bellare et al. [2] presented a formal model for this problem and a

protocol secure in the ideal-cipher model. Our work extends their work to the multi-party setting. Security proofs in the ideal-cipher model see a (keyed) cipher as a family of random permutations which are queried via an oracle to encrypt and decrypt. The oracle produces a truly random value for each new query and identical answers if the same query is asked twice; furthermore, for each key, the injectivity is satisfied. In practice, the ideal-cipher [4] is instantiated using deterministic symmetric encryption function such as AES [17]. Although these encryption functions have been designed with different criteria from being an ideal-cipher, AES has been designed with unpredictability in mind.

Security proofs in these two models together (both the random oracle and the ideal-cipher models) are superior to those provided by *ad-hoc* protocol designs although they do not provide the same security guarantees as those in the random oracle and the standard models. However, the ideal-cipher model allows for “elegant” and more efficient protocols. Boyko et al. [7,15] provided (2-party) Diffie-Hellman key exchange protocols proved secure in the random oracle model using the multi-party simulatability technique. Katz et al. [14], and Goldreich et al. [12] designed two-party key exchange protocols secure in the standard model.

Several papers have extended the Diffie-Hellman protocol [11] to the multi-party setting and thus aimed to distribute a session key among parties aggregated into a group. Bresson et al. [10] presented a formal model to securely design protocols for a scenario wherein each party holds a pair of matching public/private keys. A logical follow up to this work is a formal model for a scenario wherein the parties share a human-memorable password. This latter scenario was suggested by Asokan et al. [1] as well as protocols with informal security analysis.

## 2 Model

In this section we define a formal model for security against dictionary attacks where the adversary’s capabilities are modeled through queries. In our model, the players do not deviate from the protocol and the adversary is not a player. We define the security notion that a password-based group Diffie-Hellman protocol should achieve. In Authenticated Key Exchange (with implicit authentication), each player is assured that an adversary not in the group is unable to learn any information about the session key. Another important notion is mutual authentication, which guarantees to each player that it actually shares a session key with all the others.

### 2.1 Security Model

**Players.** We fix a nonempty set  $\mathcal{U}$  of players that can participate in a group Diffie-Hellman key exchange protocol  $P$ . A player  $U_i \in \mathcal{U}$  may have many *instances* called oracles involved in distinct, but possibly concurrent, executions of  $P$ . We denote by  $\Pi_i^t$  the  $t$ -th instance of player  $U_i$ , for any  $t \in \mathbb{N}$ .

The players share a low-entropy secret  $pw$  taken from a small dictionary Password of size  $N$ . In the following, we assume that this password  $pw$  follows a uniform distribution in the Password set.

**Abstract Interface.** Let us define the basic structure of a password-based group Diffie-Hellman protocol  $P$ . The protocol consists of two algorithms:

- The *password generation* algorithm  $\text{PWDGEN}(1^\ell)$  is a probabilistic algorithm which, on input a security parameter  $1^\ell$ , provides each player in  $\mathcal{U}$  with a common password  $pw$  uniformly distributed in  $\text{Password}$ .
- The *key exchange* algorithm  $\text{KEYEXCH}(\mathcal{U})$  is an interactive multi-party protocol providing the instances of players in  $\mathcal{U}$ , holding a common password, with a session key  $sk$ .

**Queries.** The adversary  $\mathcal{A}$  interacts with the players by making various queries. Let us explain the capability that each query captures:

- $\text{Execute}(\mathcal{U})$ : This query models passive attacks, where the adversary gets access to honest executions of  $P$  by eavesdropping. Therefore,  $\mathcal{A}$  gets back the protocol flows of an honest execution of  $P$  between the players in  $\mathcal{U}$ .
- $\text{Send}(\Pi_i^t, m)$ : This query models  $\mathcal{A}$  sending a message to an instance. The adversary  $\mathcal{A}$  gets back the response oracle  $\Pi_i^t$  generates in processing the message  $m$  according to the protocol  $P$ . A query  $\text{Send}(\Pi_1^t, \text{"Start"})$  initializes the key exchange algorithm, and thus the adversary receives the flow the first player should send out to the second one.
- $\text{Reveal}(\Pi_i^t)$ : This query models the misuse of the session key by the players. The query is only available to  $\mathcal{A}$  if oracle  $\Pi_i^t$  holds a session key. The  $\text{Reveal}$ -query unconditionally forces oracle  $\Pi_i^t$  to release  $sk_{\Pi_i^t}$  which is otherwise hidden to  $\mathcal{A}$ .
- $\text{Test}(\Pi_i^t)$ : This query models the semantic security of the session key  $sk$ . The  $\text{Test}$ -query can be asked at most once by the adversary  $\mathcal{A}$  and is only available to  $\mathcal{A}$  if  $\Pi_i^t$  is **Fresh** (see below). This query is answered as follows: one flips a coin  $b$  and forwards  $\text{Reveal}(\Pi_i^t)$  if  $b = 1$  or a random value if  $b = 0$ .

The  $\text{Execute}$ -query may at first seem useless since using the  $\text{Send}$ -query the adversary has the ability to carry out honest executions of  $P$  among parties. Yet the  $\text{Execute}$ -query is essential for properly dealing with dictionary attacks. The number  $q_s$  of  $\text{Send}$ -queries directly asked by the adversary does not take into account the number of  $\text{Execute}$ -queries. Therefore,  $q_s$  represents the number of flows the adversary may have built by himself, and thus the number of passwords he would have tried.

The security notions take place in the context of executing  $P$  in the presence of the adversary  $\mathcal{A}$ . In this game  $\text{Game}^{\text{ake}}(\mathcal{A}, P)$ ,  $\mathcal{A}$  plays against the players using the above queries in order to defeat the security of  $P$ . The game is initialized by providing coin tosses to  $\text{PWDGEN}$ ,  $\mathcal{A}$ , all  $\Pi_i^t$ , and then

1.  $\text{PWDGEN}$  is run to set the value  $pw$  of the password,
2. Initialize any  $\Pi_i^t$  with  $sk_{\Pi_i^t} \leftarrow \text{NULL}$ ,
3. Initialize adversary  $\mathcal{A}$  with  $1^\ell$  and access to all  $\Pi_i^t$ ,
4. Run adversary  $\mathcal{A}$  and answer queries made by  $\mathcal{A}$ ,
5. At the end of the game,  $\mathcal{A}$  outputs its guess  $b'$  for the bit  $b$  involved in the  $\text{Test}$ -query.



2.2 Security Notions

**Freshness.** An oracle  $\Pi_i^t$  is **Fresh** (or holds a **Fresh** key  $sk$ ) if  $\Pi_i^t$  has computed a session key  $sk \neq \text{NULL}$  and neither  $\Pi_i^t$  nor one of its partners has been asked for a **Reveal**-query. Intuitively, the partners of an instance  $\Pi_i^t$  are all the instances that “should” hold the same session key as  $\Pi_i^t$  at the end of the protocol. We give a more formal definition in the full version of this paper [8].

**AKE Security.** In an execution of  $P$ , we say an adversary  $\mathcal{A}$  *wins* if it asks a single **Test**-query to a **Fresh** player  $U$  and correctly guesses the bit  $b$  used in the game  $\text{Game}^{\text{ake}}(\mathcal{A}, P)$ . We denote the **AKE advantage** as  $\text{Adv}_P^{\text{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$ , where the probability space is over all the random coins of the adversary and all the oracles.

3 Assumptions

Before presenting the protocol, let us remind the algorithmic assumptions on which its security will be based on. These assumptions were shown in [9] to be reasonable by relating them to the DDH and CDH.

Let  $\mathbb{G} = \langle g \rangle$  be a cyclic group of prime order  $q$  and  $n \in \mathbb{N}$ . Let  $I_n$  be  $\{1, \dots, n\}$ ,  $\mathcal{P}(I_n)$  be the set of all subsets of  $I_n$  and  $\Gamma$  be any subset of  $\mathcal{P}(I_n)$ . We define the *Group Diffie-Hellman distribution* relative to  $\Gamma$  as:

$$\text{GDH}_\Gamma = \{ \mathcal{D}_\Gamma(x_1, \dots, x_n) \mid x_1, \dots, x_n \in_R \mathbb{Z}_q \},$$

where

$$\mathcal{D}_\Gamma(x_1, \dots, x_n) = \left\{ \left( J, g^{\prod_{j \in J} x_j} \right) \mid J \in \Gamma \right\}.$$

Our protocol in this paper is based on the triangular structure  $\mathcal{T}_n$  for  $\Gamma$ , we illustrate for  $n = 4$  on Figure 1:

$$\begin{aligned} \mathcal{T}_n &= \bigcup_{2 \leq j \leq n} \{ \{i \mid 1 \leq i \leq j, i \neq k\} \mid 1 \leq k \leq j \} \\ &= \{ \{ \}; \{2\}, \{1\}; \{2, 3\}, \{1, 3\}, \{1, 2\}; \{2, 3, 4\}, \{1, 3, 4\}, \{1, 2, 4\}, \{1, 2, 3\}; \dots \}. \end{aligned}$$

$g$			
$g^{x_2}$	$g^{x_1}$		
$g^{x_2 x_3}$	$g^{x_1 x_3}$	$g^{x_1 x_2}$	
$g^{x_2 x_3 x_4}$	$g^{x_1 x_3 x_4}$	$g^{x_1 x_2 x_4}$	$g^{x_1 x_2 x_3}$

Fig. 1. Trigon defined by  $\mathcal{T}_n$  when  $n = 4$ .

**Trigon Group Computational Diffie-Hellman Assumption (TG-CDH).**

A  $(T, \varepsilon)$ -TG-CDH $_n$ -attacker for  $\mathbb{G}$  is a probabilistic Turing machine  $\Delta$  running in time  $T$  that given  $\mathcal{D} = \mathcal{D}_{\mathcal{T}_n}(x_1, \dots, x_n) \in \text{GDH}_{\mathcal{T}_n}$  outputs  $g^{x_1 \cdots x_n}$  with probability greater than  $\varepsilon$ . We denote this success probability  $\text{Succ}_{\mathbb{G}}^{\text{tgcdh}_n}(\Delta)$ .

**Multi Decisional Diffie-Hellman Assumption (M-DDH).** In the analysis of the protocol EKE we need an equivalent version of the DDH assumption. Let us define the two following distributions:

$$\begin{aligned} \text{M-DH}_n &= \{(g^{x_1}, \dots, g^{x_n}, g^{r x_1}, \dots, g^{r x_n}) \mid x_1, \dots, x_n, r \in_R \mathbb{Z}_q\}, \\ \text{Rand}_n &= \{(g^{x_1}, \dots, g^{x_n}, g^{y_1}, \dots, g^{y_n}) \mid x_1, \dots, x_n, y_1, \dots, y_n \in_R \mathbb{Z}_q\}. \end{aligned}$$

A  $(T, \varepsilon)$  - M-DDH $_n$ -distinguisher for  $\mathbb{G}$  is a probabilistic Turing machine  $\Delta$  running in time  $T$  that is able to distinguish the two distributions with advantage  $\text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(\Delta)$  greater than  $\varepsilon$ .

**Lemma 1.** *For any group  $\mathbb{G}$  and any integer  $n$ ,  $\text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(T) \leq (n-1) \text{Adv}_{\mathbb{G}}^{\text{ddh}}(T)$  and  $\text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(T) \leq \text{Adv}_{\mathbb{G}}^{\text{ddh}}(T + (4n-6)\tau_{\mathbb{G}})$ , where  $\tau_{\mathbb{G}}$  is the computational time for an exponentiation in  $\mathbb{G}$ .*

*Proof.* The first result easily comes using a hybrid argument [16], while the second one uses the random self-reducibility. Indeed, from a decisional Diffie-Hellman instance  $(g^{x_2}, g^{r_1}, g^{r_2 x_2})$ , where  $r_2 = r_1$ , one derives (with 2 exponentiations performed by raising two values to the power of  $x_1$ ) a 4-tuple  $(A_1 = g^{x_1}, A_2 = g^{x_2}, B_1 = g^{r_1 x_1}, B_2 = g^{r_2 x_2})$ . Then, one easily gets a  $2n$ -tuple  $(A_1 = g^{x_1}, \dots, A_n = g^{x_n}, B_1 = g^{r_1 x_1}, \dots, B_n = g^{r_n x_n})$  where either all the  $r_i$  are equal (if  $r_2 = r_1$ ), or the  $r_i$  are independent from one another (if  $r_2 \neq r_1$ ). To this aim, one chooses a random pair  $(u_i, v_i)$ , and computes

$$\begin{aligned} A_i &= A_1^{u_i} A_2^{v_i} = g^{x_1 u_i + x_2 v_i} = g^{x_i} \\ B_i &= B_1^{u_i} B_2^{v_i} = g^{r_1 x_1 u_i + r_2 x_2 v_i} = g^{r_1 x_i + (r_2 - r_1) x_2 v_i}. \end{aligned}$$

□

## 4 A Password-Based Group Diffie-Hellman Protocol

In the following theorems and proofs we assume both the random oracle model and the ideal-cipher model, and the arithmetic is in a finite cyclic group  $\mathbb{G} = \langle g \rangle$  of order a  $\ell_1$ -bit prime number  $q$ , where the operation is denoted multiplicatively. More precisely, we consider  $\bar{\mathbb{G}} = \mathbb{G} \setminus \{1\}$ . It has the particularity that for any  $h \in \bar{\mathbb{G}}$ ,  $\bar{\mathbb{G}} = \{h^r \mid r \in \{1, \dots, q-1\}\}$ , but it is no longer a group.

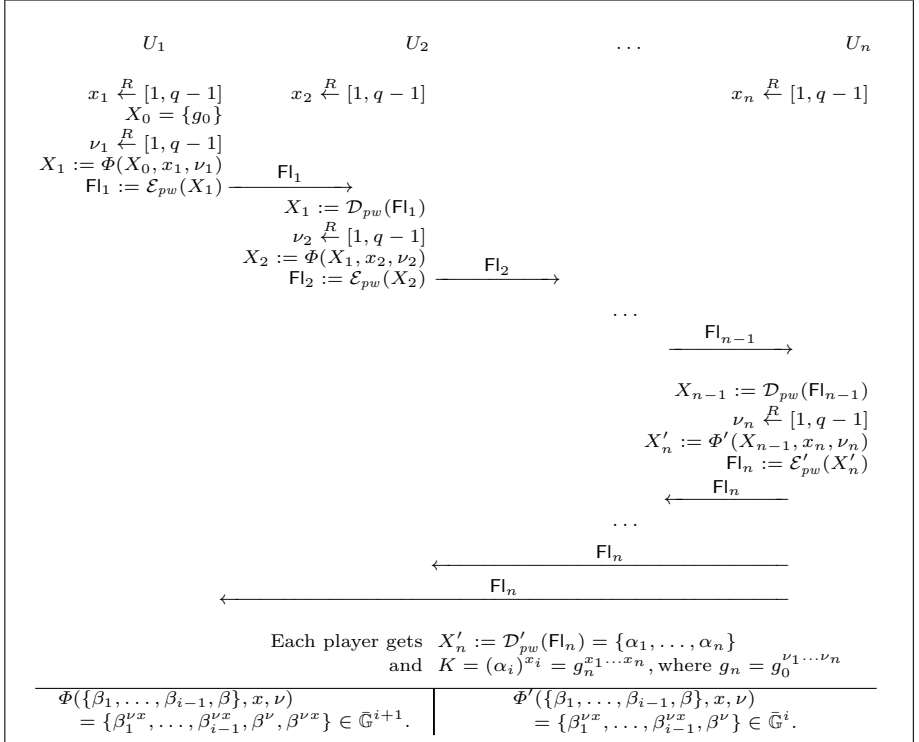
We then use a hash function  $\mathcal{H}$  from  $\{0, 1\}^*$  to  $\{0, 1\}^{\ell_2}$  and consider several block ciphers, depending on the size of the input: for each integer  $i \geq 2$ , we define two families  $\mathcal{E}^i = \{\mathcal{E}_k^i\}$  and  $\mathcal{E}'^i = \{\mathcal{E}'_k^i\}$  of keyed permutations over  $\bar{\mathbb{G}}^i$ , where  $k \in \text{Password}$ . The inverse of  $\mathcal{E}_k^i$  (resp.  $\mathcal{E}'_k^i$ ) is denoted  $\mathcal{D}_k^i$  (resp.  $\mathcal{D}'_k^i$ ).

In practice, such encryption schemes are instantiated with CBC mode so that each part of the plaintext depends on the entire ciphertext. Following this idea, we (abusively) denote  $\mathcal{E}_k(X)$  (resp.  $\mathcal{E}'_k(X)$ ) the encryption of a plaintext  $X \in \mathbb{G}^i$  for some  $i$  under key  $k$  using  $\mathcal{E}_k^i$  (resp.  $\mathcal{E}'_k^i$ ) without explicitly specifying the length of  $X$ .

#### 4.1 Algorithm

As illustrated on Figure 2, the protocol EKE consists of a set of players arranged in a ring and the flows are encrypted under the password  $pw$ . The session-key space **SK** associated to this protocol is  $\{0,1\}^{\ell_2}$  equipped with a uniform distribution. Moreover, EKE consists of two stages: several up-flows (which are encrypted using  $\mathcal{E}$ ) and the down-flow (which is encrypted using  $\mathcal{E}'$ ).

In the up-flow, player  $U_i$  (for  $1 \leq i < n$ ) receives a ciphertext  $\text{Fl}_{i-1} \in \mathbb{G}^i$  and decrypts it using  $\mathcal{D}_{pw}$  into the plaintext  $X_{i-1} \in \mathbb{G}^i$  (by convention,  $U_1$  just receives  $\text{Fl}_0 = \text{"Start"}$ , and thus builds  $X_0 = \{g_0\}$ , where  $g_0$  is a random element in  $\mathbb{G}$ ). Player  $U_i$  then generates at random two (private) values  $(x_i, \nu_i)$  in  $\mathbb{Z}_q^*$  and gets  $X_i := \Phi(X_{i-1}, x_i, \nu_i) \in \mathbb{G}^{i+1}$  by processing the plaintext  $X_{i-1}$



**Fig. 2.** Protocol EKE. The multicast group is  $\mathcal{U} = \{U_1, U_2, \dots, U_n\}$  and the session key is  $sk = \mathcal{H}(\mathcal{U} \parallel \text{Fl}_n \parallel K)$ .

according to the operator  $\Phi$  (described below). Player  $U_i$  finally encrypts the value  $X_i$  using  $\mathcal{E}_{pw}$  and forwards the ciphertext  $\text{Fl}_i$  to the next player in the ring.

The down-flow takes place when player  $U_n$  receives the last up-flow  $\text{Fl}_{n-1} \in \bar{\mathbb{G}}^n$ . It decrypts it using  $\mathcal{D}_{pw}$  into the plaintext  $X_{n-1} \in \bar{\mathbb{G}}^n$ . It then generates at random two (private) values  $(x_n, \nu_n)$  in  $\mathbb{Z}_q^*$  and gets  $X'_n := \Phi'(X_{n-1}, x_n, \nu_n) \in \bar{\mathbb{G}}^n$  by processing the plaintext  $X_{n-1}$  according to the operator  $\Phi'$  (described below). Player  $U_n$  finally encrypts the value  $X'_n$  using  $\mathcal{E}'_{pw}$  and broadcasts the ciphertext  $\text{Fl}_n$ .

Finally, each player can compute the session key  $sk = \mathcal{H}(\mathcal{U} \parallel \text{Fl}_n \parallel K)$ , where  $K = (g_0^{\nu_1 \cdots \nu_n})^{x_1 \cdots x_n}$ . Indeed, if everything worked correctly, player  $U_i$  can compute  $K$  by decrypting the broadcast  $\text{Fl}_n$  using  $\mathcal{D}'_{pw}$  into  $X'_n \in \bar{\mathbb{G}}^n$  and raising the  $i$ -th term  $\alpha_i$  of  $X'_n$  to the power of its private exponent  $x_i$ .

## 4.2 Operators $\Phi$ and $\Phi'$

We now describe the operators  $\Phi$  and  $\Phi'$ , and see that finally all the players agree on the same value  $K$ . The operator  $\Phi$  takes as inputs a set  $\{\beta_1, \dots, \beta_{i-1}, \beta\} \in \bar{\mathbb{G}}^i$ , for some  $i$ , a private exponent  $x \in \mathbb{Z}_q^*$  and a blinding exponent  $\nu \in \mathbb{Z}_q^*$ . Then,

$$\Phi(\{\beta_1, \dots, \beta_{i-1}, \beta\}, x, \nu) = \{\beta_1^{\nu x}, \dots, \beta_{i-1}^{\nu x}, \beta^\nu, \beta^{\nu x}\} \in \bar{\mathbb{G}}^{i+1}.$$

The operator  $\Phi'$  does exactly the same transformation but returns the  $i$  first elements only:

$$\Phi'(\{\beta_1, \dots, \beta_{i-1}, \beta\}, x, \nu) = \{\beta_1^{\nu x}, \dots, \beta_{i-1}^{\nu x}, \beta^\nu\} \in \bar{\mathbb{G}}^i.$$

Therefore, if all the computations are performed correctly, the flows between 4 players include the plaintexts  $X_1$ ,  $X_2$  and  $X_3$  presented on Figure 3. The plaintext  $X'_4$  is the 4 first elements of  $X_4$  only while the last element of  $X_4$  is  $K = (g_0^{\nu_1 \nu_2 \nu_3 \nu_4})^{x_1 x_2 x_3 x_4} = g_4^{x_1 x_2 x_3 x_4}$ .

One can indeed check by induction that the  $j$ -th element of  $X_i$  is  $(g_0^{\mu_i})^{y_i/x_j} = g_i^{y_i/x_j}$ , where  $\mu_i = \nu_1 \cdots \nu_i \bmod q$ ,  $g_i = g_0^{\mu_i}$  and  $y_i = x_1 \cdots x_i \bmod q$ . Therefore, the  $i$ -th element  $\alpha_i$  of the down-flow is  $g_n^{y_n/x_i}$  which with the knowledge of  $x_i$  leads to the common value  $K = \alpha_i^{x_i} = g_n^{y_n}$ .

$g_0$					$= X_0$
$g_1$	$g_1^{x_1}$				$= X_1 = \Phi(X_0, x_1, \nu_1)$
$g_2^{x_2}$	$g_2^{x_1}$	$g_2^{x_1 x_2}$			$= X_2 = \Phi(X_1, x_2, \nu_2)$
$g_3^{x_2 x_3}$	$g_3^{x_1 x_3}$	$g_3^{x_1 x_2}$	$g_3^{x_1 x_2 x_3}$		$= X_3 = \Phi(X_2, x_3, \nu_3)$
$g_4^{x_2 x_3 x_4}$	$g_4^{x_1 x_3 x_4}$	$g_4^{x_1 x_2 x_4}$	$g_4^{x_1 x_2 x_3}$	$g_4^{x_1 x_2 x_3 x_4}$	$= X_4 = \Phi(X_3, x_4, \nu_4)$

**Fig. 3.** Honest execution, when  $n = 4$ . We denote  $\nu_i = \log_{g_{i-1}}(g_i)$ .

### 4.3 Dictionary Attacks

In EKE, we have to be careful of the content in the ciphertext since any redundancy in the concatenation of the plaintexts in the flows of the protocol could be used by the adversary to mount a dictionary attack. The adversary could decrypt flows using all the passwords in the dictionary and look for this redundancy.

Namely, the trivial conversion wherein one substitutes in a group Diffie-Hellman protocol [10] the signature scheme by a symmetric encryption scheme is easily seen insecure, while it works in the two-party case. This conversion indeed produces a protocol in which all the computations are performed with  $\nu_i = 1$ , for all  $i$ ; therefore the last element of each plaintext in  $\text{Fl}_i$  also belongs to the plaintext in  $\text{Fl}_{i+1}$ .

## 5 Security Result

In this section we assert that under reasonable and well-defined intractability assumptions the protocol EKE securely distributes session keys. We deal with the AKE goal only and thus do not consider forward-secrecy here. However, concurrent executions are possible.

**Theorem 1.** *Let  $P$  be the EKE protocol,  $\mathbf{SK}$  be the session-key space and Password be a finite dictionary of size  $N$ . Let  $\mathcal{A}$  be an adversary against the AKE security of  $P$  within a time bound  $T$ , after  $q_s$  interactions with the parties,  $q_h$  hash-queries, and  $q_e$  encryption/decryption queries. Then we have:*

$$\text{Adv}_P^{\text{ake}}(\mathcal{A}) \leq \frac{2q_s}{N} + 2q_s \text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(T') + 2q_h \text{Succ}_{\mathbb{G}}^{\text{tgcdh}_n}(T') + \frac{2Q^2}{q^2}$$

where  $T' \leq T + nQ\tau_{\mathbb{G}}$ ,  $Q = 3q_s + q_e$  and  $\tau_{\mathbb{G}}$  is the computational time for an exponentiation in  $\mathbb{G}$ . (Recall that  $q$  is the order of  $\mathbb{G}$ ).

This theorem shows that EKE is secure against dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions (number of Send-queries) to the number of passwords. This is particularly significant in practice since a password may expire once a number of failed interactions has been achieved, whereas the adversary's capability to enumerate passwords off-line is only limited by its computational power.

Of course, the security results only holds provided that the adversary does not solve either the trigon group computational problem TG-CDH or the multi-decisional Diffie-Hellman problem M-DDH. But these terms can be made negligible by appropriate choice of parameters for the group  $\mathbb{G}$ .

## 6 Proof of Security

In this section we show that the protocol EKE achieves security against dictionary attacks as claimed by Theorem 1. We first introduce the notations we will

use and then prove that the best the adversary can do is to essentially eliminate one password from the dictionary per initiated session (maybe concurrently). Here, we present a proof that does not yet deal with forward-secrecy. All the lemmas are proven in the full version of this paper [8].

## 6.1 Operator $\Psi$

As illustrated in Figure 2, each player  $U_i$  generates a new basis when processing an up-flow by raising the values it received to the power of its random blinding exponent  $\nu_i$ . But let us notice that given a TG-CDH instance of size  $n$ , with the TG-CDH-solution, one can easily derivate the trigon whose lines are the flows sent during an honest execution of EKE, by raising the lines to the power of random and independent exponents.

We denote by  $\theta$  a  $n$ -tuple of elements in  $\mathbb{Z}_q^*$ , by  $\theta_i$  the  $i$ -th component of  $\theta$ , and by  $[g]_n$  the  $n$ -tuple  $(g, \dots, g)$ . For any line  $L$  of length  $i + 1$ , the operator  $\Psi$  takes as input a  $n$ -tuple  $\theta$ , a random exponent  $\nu$  and applies a (multiplicative) self-reduction of the line  $L$  as follows:

- **[Using  $\theta$ ]** first, one raises the first  $i$  elements of  $L$  to the power of  $\Theta_i/\theta_1, \dots, \Theta_i/\theta_i$  respectively, and the last element to the power of  $\Theta_i$ , where  $\Theta_i = \theta_1 \cdots \theta_i$ ;
- **[Change of Basis]** then, one raises all the elements of the tuple to the power  $\nu$ .

For example, from a line  $L = (g_1, \dots, g_{i+1})$ , with any tuple  $\theta$  and  $\nu$ , one gets

$$\Psi(L, \theta, \nu) = (g_1^{\nu\Theta_i/\theta_1}, \dots, g_i^{\nu\Theta_i/\theta_i}, g_{i+1}^{\nu\Theta_i}),$$

where  $\Theta_i = \theta_1 \cdots \theta_i$ . A line  $L$  of form  $\{g^{y_i/x_1}, \dots, g^{y_i/x_i}, g^{y_i}\}$  where  $y_i = x_1 \cdots x_i$  is thus represented as follows:

$$L = \Psi([g]_{i+1}, (x_1, \dots, x_i, 1, \dots, 1), 1) \in \bar{\mathbb{G}}^{i+1}.$$

The following lemmas exhibit some useful results about the operators  $\Phi$  and  $\Psi$ :

**Lemma 2 (Equality of Distributions).** *Let  $g \in \bar{\mathbb{G}}$  and  $L = \{g^{\alpha_0}, \dots, g^{\alpha_i}\} \in \bar{\mathbb{G}}^{i+1}$ . The following two distributions are perfectly indistinguishable:*

$$\left\{ \Psi(L, \theta, \nu) \right\}_{(\theta, \nu) \in (\mathbb{Z}_q^*)^{n+1}} \quad \text{and} \quad \left\{ g^{r_0}, \dots, g^{r_i} \right\}_{(r_0, \dots, r_i) \in (\mathbb{Z}_q^*)^{i+1}}$$

**Lemma 3 (Commutativity and Composition).** *Let  $x, \nu, \nu' \in \mathbb{Z}_q^*$  and  $\theta, \theta' \in (\mathbb{Z}_q^*)^n$ . For any line  $L \in \bar{\mathbb{G}}^i$ , we have (where  $\theta\theta'$  is the component-wise multiplication of the vectors  $\theta$  and  $\theta'$ ):*

$$\begin{aligned} \Psi(\Phi(L, x, \nu), \theta, \nu') &= \Phi(\Psi(L, \theta, \nu'), x\theta_i, \nu); \\ \Psi(\Psi(L, \theta, \nu), \theta', \nu') &= \Psi(L, \theta\theta', \nu\nu'); \\ \Psi(\Phi(L, x, \nu), [1]_n, \nu') &= \Phi(L, x, \nu\nu'); \\ \text{if } (\forall j < i, \theta_j &= \theta'_j), \Psi(L, \theta, \nu) = \Psi(L, \theta', \nu). \end{aligned}$$

## 6.2 Proof of Theorem 1

In this section we incrementally define a sequence of games starting at the real game  $\mathbf{G}_0$  and ending up at  $\mathbf{G}_6$ . We let  $b$  and  $b'$  be defined as in Section 2.1 and refer to  $\mathbf{S}_i$  as the event  $b = b'$  in game  $\mathbf{G}_i$ . We also define the event  $\mathbf{Encrypt}_i$  as the event that a flow has been encrypted, but not decrypted *first* (see below), by the adversary under  $pw$  (with any symmetric encryption scheme  $\mathcal{E}$  or  $\mathcal{E}'$ ). We use the following lemma within our sequence of games [20]:

**Lemma 4.** *Let  $E, E'$  and  $F, F'$  be some events defined on a probability space. Let us assume that  $\Pr[F] = \Pr[F'] = \varepsilon$  and  $\Pr[E \wedge \neg F] = \Pr[E' \wedge \neg F']$ . Then,  $|\Pr[E] - \Pr[E']| \leq \varepsilon$ .*

**Game  $\mathbf{G}_0$ :** This is the real attack  $\mathbf{Game}^{\text{ake}}(\mathcal{A}, P)$  in which several oracles are available to the adversary: a hash oracle, the encryption/decryption oracles, and all the instances of players (in order to cover concurrent executions).

**Rule 1:** The instances of players process each **Send**-query with a pair of random exponents  $(x_i, \nu_i)$ , using the operators  $\Phi$  and  $\Phi'$ .

Thus, the instances of players can easily answer to the **Reveal**-query and the **Test**-query. The **Execute**-query is proceeded similarly. By definition,  $\Pr[\mathbf{S}_0] = (\text{Adv}_P^{\text{ake}}(\mathcal{A}) + 1)/2$ .

**Game  $\mathbf{G}_1$ :** We simulate the hash and the encryption/decryption oracles as in  $\mathbf{G}_0$  by maintaining five lists: a hash list  $(\Lambda_H)$ , encryption lists  $(\Lambda_E, \Lambda'_E)$  and decryption lists  $(\Lambda_D, \Lambda'_D)$ . The lists are initially empty. We denote by  $q_H$  the size of  $\Lambda_H$ , and by  $q_E$  the number of encryption-decryption relations: i.e.  $q_E$  is the size of  $\Lambda_E \cup \Lambda'_E$ . The queries are answered as follows:

- *Hash-query:* For a query  $q$  such that a record  $(q, r)$  appears in  $\Lambda_H$ , the answer is  $r$ . Otherwise  $r$  is chosen at random from  $\{0, 1\}^{\ell_2}$  and the record  $(q, r)$  is added to  $\Lambda_H$ . We have  $\mathcal{H}(q) = r$ .
- *Encryption-query:* For an encryption query  $(k, X)$  to  $\mathcal{E}$  (resp.  $\mathcal{E}'$ ) such that a record  $(*, k, X, Y)$  appears in  $\Lambda_E$  (resp.,  $\Lambda'_E$ ) the answer is  $Y$ . Otherwise  $Y$  is a random ciphertext of length  $|X|$ . The record  $(\delta, k, X, Y)$  is then added to encryption list  $\Lambda_E$  (resp.  $\Lambda'_E$ ).  $\delta \in \{0, 1\}$  is a bit indicating the originator of the query: if the query comes from the simulator then  $\delta = 0$  else the query comes from the adversary  $\delta = 1$ .
- *Decryption-query:* For a decryption query  $(k, Y)$  to  $\mathcal{D}$  (resp.  $\mathcal{D}'$ ) such that a record  $(*, k, X, Y)$  appears in  $\Lambda_E$  (resp.  $\Lambda'_E$ ), the answer is  $X$ . Otherwise  $X$  is generated by the following rule:

**Rule 2:**  $X$  is a random tuple  $\{g^{r_i}\}_{1 \leq i \leq |Y|}$  where  $r_1, \dots, r_{|Y|}$  are randomly drawn in  $\mathbb{Z}_q^*$ .

The record  $(k, Y, X)$  is then added to the decryption list  $\Lambda_D$  (resp.  $\Lambda'_D$ ) while the record  $(0, k, X, Y)$  is added to the encryption list  $\Lambda_E$  (resp.  $\Lambda'_E$ ).

We notice that a decryption-query adds a record to both the encryption and the decryption lists, while an encryption-query adds a record to the encryption list only. In both cases, a later encryption or decryption query on the same elements does not add any record to any list. Hence, a record  $(k, Y, X)$  appears in the decryption list if and only if the decryption query  $(k, Y)$  has been asked *first* (i.e., before the corresponding encryption query).

With this definition,  $\text{Encrypt}_i$  is defined in game  $\mathbf{G}_i$  as the event that there exists a record  $(1, pw, X, Y)$  in an encryption list, such that  $Y$  has been submitted in a  $\text{Send}$ -query. Note that this implies that the corresponding record  $(pw, Y, X)$  does not appear in any decryption list.

From the above simulation we easily see that the games  $\mathbf{G}_1$  and  $\mathbf{G}_0$  are perfectly indistinguishable, unless the permutation property of the block ciphers does not hold. One could have avoided collisions in the above simulation but this is at most  $q_E^2/2(q-1)^2$  since the smallest set for the encryption functions is  $|\bar{\mathcal{G}}^2| = (q-1)^2$ :

$$|\Pr[S_1] - \Pr[S_0]| \leq \frac{q_E^2}{2(q-1)^2} \leq \frac{q_E^2}{q^2}. \quad (1)$$

**Game  $\mathbf{G}_2$ :** We delete the executions wherein the adversary may have guessed the password. More formally, we delete the executions wherein event  $\text{Encrypt}_1$  occurs: i.e. a  $\text{Send}(\Pi, Y)$ -query is asked and a record of the form  $(1, pw, *, Y)$  appears in an encryption list. In these executions, we stop setting  $b'$  at random:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\text{Encrypt}_1]. \quad (2)$$

**Game  $\mathbf{G}_3$ :** We simulate the instances of players from a tuple  $(x_1, \dots, x_n)$ . This tuple allows us to compute an instance  $\mathcal{D} = \mathcal{D}(x_1, \dots, x_n)$  of the  $\text{TG-CDH}_n$  with its solution  $g^{x_1 \cdots x_n}$ . We use  $\mathcal{D}$  to construct using blinding exponents  $\nu_1, \dots, \nu_n$  the triangular structure illustrated in Figure 3 where all the bases are randomized. The lines of this structure will be used in this game to answer to the  $\text{Send}$ -queries. The lines of this triangular structure are denoted and constructed as

$$L_i = \begin{cases} \{g\} & \text{if } i = 0, \\ \Phi(L_{i-1}, x_i, \nu_i) & \text{if } 1 \leq i \leq n, \end{cases}$$

where the  $n+1$ -th element  $\kappa$  of  $L_n$  is  $\kappa = g_n^{x_1 \cdots x_n} = (g^{x_1 \cdots x_n})^{\nu_1 \cdots \nu_n}$ .

We now show how to use these lines to simulate the instances of players. We first maintain a list  $\Lambda_\Psi$  that keeps track of the exponents  $\theta$  used to blind a line  $L_i$ :  $L = \Psi(L_i, \theta, \nu)$ . This list contains records of the form  $(i, \theta, \nu, L)$  and is initially set to  $\{(0, [1]_n, 1, \{g\})\}$ . Then, we answer to a  $\text{Send}(\Pi_i^t, \text{Fl})$ -query as follows:

- $\Pi_i^t$  is waiting for an up-flow: if the length of  $\text{Fl}$  is different from  $i$ , then we do not do anything. Otherwise we do perform the following two steps.
  1. if  $i = 1$  then one sets  $L = \{g\}$ , else one invokes the decryption oracle to get  $L = \mathcal{D}_{pw}(\text{Fl})$ .



2. one computes line  $L'$  according to **Rule 1.1** and encrypts *some* elements of  $L'$ . If  $i < n$ , then the whole line  $L'$  is encrypted into  $\mathcal{E}_{pw}(L')$ . Otherwise, if  $i = n$ , then only the  $n$  first elements of  $L'$ , we refer to them as  $L''$ , are encrypted using  $\mathcal{E}'_{pw}$ . Finally,  $\Pi_i^t$  waits for the down-flow.

**Rule 1.1:** We first chooses two random exponents  $(\rho_i^t, \mu_i^t) \in (\mathbb{Z}_q^*)^2$ . If  $(i-1, \theta, \nu, L) \in A_\Psi$  for some  $\theta, \nu$ , then we compute  $L' = \Psi(L_i, \theta', \mu_i^t \nu)$  where  $\theta'$  is defined to  $\theta$  except that  $\theta'_i = \rho_i^t$ , and we update the list  $A_\Psi$ . Otherwise one applies the **Rule 1.1'** presented below.

**Rule 1.1':** One still uses **Rule 1**, but with  $(x_i \rho_i^t, \nu_i \mu_i^t)$  instead of  $(x_i, \nu_i)$ .

The random  $\rho_i^t$  is different each time one answers this flow (either by **Rule 1.1** or **Rule 1.1'**.) Indeed, the same flow  $\text{Fl}$  may be sent several times by the adversary in different and concurrent executions.

By induction, one easily show that any line  $L$  either comes from **Rule 2** or, under  $\neg\text{Encrypt}$ , from a previous **Rule 1.1**.

- $\Pi_i^t$  is waiting for a down-flow: if the length of  $\text{Fl}$  is different from  $n$ , then we do not do anything. Otherwise, we invoke the decryption oracle  $\mathcal{D}'_{pw}(\text{Fl})$  to obtain  $L''$  or more specifically to obtain the  $i$ -th element  $\alpha_i^t$  in  $L''$ :

**Rule 3:** For any  $\Pi_i^t$ ,  $K_i^t$  is set to be  $(\alpha_i^t)^{x_i \rho_i^t}$ , where  $\alpha_i^t$  is the  $i$ -th element in the down-flow  $L''$  received by  $\Pi_i^t$ .

We have now to show that the  $\Phi$  relation between the lines  $L_{i-1}$  and  $L_i$  is “preserved” by the  $\Psi$  transformation. The following lemma shows it: two lines  $L_{i-1}, L_i$  of the triangular structure already related by the operator  $\Phi$  are still related by  $\Phi$  after having respectively been transformed into  $L, L'$  by the operator  $\Psi$ .

**Lemma 5.**  $L' = \Phi(L, x_i \rho_i^t, \mu_i^t \nu_i)$ .

By Lemma 5, our simulation simply makes the player choose as private exponent  $x_i \rho_i^t$  and as blinding exponent  $\nu_i \mu_i^t$  (both values are uniformly distributed because  $\mu_i^t$  and  $\rho_i^t$  are). From the value  $K_i^t$ , we can then easily compute the session key  $sk_{\Pi_i^t}$  to be  $\mathcal{H}(\mathcal{U} \parallel \text{Fl}_n \parallel K_i^t)$ .

It follows that games  $\mathbf{G}_2$  and  $\mathbf{G}_3$  are perfectly indistinguishable:

$$\Pr[\mathbf{S}_3] = \Pr[\mathbf{S}_2]. \quad (3)$$

**Game  $\mathbf{G}_4$ :** We now modify the way the decryption queries are simulated by modifying the **Rule 2**, in order to embed the instance  $\mathcal{D}$  in the answers output by the decryption oracle, so that an attack may help us to solve it.

**Rule 2.1:** One chooses a random blinding exponent  $\nu \in \mathbb{Z}_q^*$  and random exponents  $\theta$  in  $(\mathbb{Z}_q^*)^n$ . If  $Y$  is a query to  $\mathcal{D}_{pw}$ , then  $X$  is set to  $\Psi(L_i, \theta, \nu)$ , where  $i = |Y| - 1$ . If  $Y$  is a query to  $\mathcal{D}'_{pw}$ , and  $|Y| = n$ , then  $X'$  is set to  $\Psi(L_n, \theta, \nu)$ , but  $X$  is set to the  $n$  first elements of  $X'$ . In both cases, the list  $A_\Psi$  is updated.

From Lemma 2 with random  $\theta$  and  $\nu$ , all the answers  $X$  are perfectly random in  $\mathbb{G}^{|Y|}$ :

$$\Pr[S_4] = \Pr[S_3]. \quad (4)$$

Before going further on, let us claim the following lemma. It shows that from now on, **Rule 1.1'** will not be used anymore.

**Lemma 6.** *If one assumes  $\neg\text{Encrypt}_4$  in game  $\mathbf{G}_4$ , any plaintext  $L$  included in a flow received via a **Send-query** is recorded in  $\Lambda_\Psi$  (possibly with one more element if  $L$  has been decrypted by  $\mathcal{D}'_{pw}$ , and thus corresponds to a down-flow).*

**Game  $\mathbf{G}_5$ :** In the above game  $\mathbf{G}_4$ , one can remark that knowledge of the  $x_i$ 's is not needed, one could only be given an instance  $\mathcal{D} = \mathcal{D}(x_1, \dots, x_n)$  of the  $\text{TG-CDH}_n$  with its solution  $g^{x_1 \cdots x_n}$  only.

But while the  $x_i$ 's are not needed to construct the triangular structure  $\{L_0, \dots, L_n\}$ , the  $x_i$ 's are needed to compute  $K_i^t$ . This value is in turn used to compute the session key  $sk_{\Pi_i^t}$  and therefore needed to answer to the **Reveal-query** and **Test-query**. Thus, if we want to avoid the use of the  $x_i$ 's, we need to find another way to compute  $K_i^t$ .

Fortunately, it is possible: according to Lemma 6 if  $\text{Encrypt}_4$  did not occur then the upflow  $L' = \Phi(L, x_i \rho_i^t, \nu_i \mu_i^t)$  has been generated by the **Rule 1.1** with as input of  $L = \Psi(L_{i-1}, \theta, \nu)$ . Let us recall that  $L''$  corresponds to the  $n$  first elements of the tuple  $\Psi(L_n, \theta', \nu')$ ,  $\kappa = g^{x_1 \cdots x_n}$  is the last component of  $L_n$  and  $\theta'$  is equal to the product  $\theta'_1 \cdots \theta'_n$ . We can now compute  $K_i^t$  by modifying **Rule 3** to be:

**Rule 3.1:**  $K_i^t = \kappa^{\nu' \theta' \cdot \rho_i^t / \theta'_i}$ . And then, the session keys can be computed without the  $x_i$ 's.

**Lemma 7.** *Rule 3 and Rule 3.1 lead to the same result.*

By Lemma 7, the games  $\mathbf{G}_4$  and  $\mathbf{G}_5$  are perfectly indistinguishable, as soon as  $\text{Encrypt}_5$  does not occur.

$$\Pr[S_5] = \Pr[S_4]. \quad (5)$$

**Game  $\mathbf{G}_6$ .** Finally, we are just given an instance  $\mathcal{D} = \mathcal{D}(x_1, \dots, x_n)$  of the  $\text{TG-CDH}_n$  without its solution  $g^{x_1 \cdots x_n}$ . Then, if the adversary helps us to get some  $K_i^t$ , we have solved the  $\text{TG-CDH}_n$  problem (and we are done, see the Lemma 8 below).

However, since we do not know  $\kappa$ , we can no longer compute  $K_i^t$  and can not therefore answer to **Reveal-oracles** (and the **Test-query**). We simply simulate the **Reveal-oracles** (even for a **Test-query**), by answering a random value, without asking the hash oracle  $\mathcal{H}$ .

Let us denote by **AskH** the event that  $\mathcal{A}$  makes a hash-query of the form  $(\mathcal{U} \parallel \text{Fl}_n \parallel K_i^t)$ , where  $\text{Fl}_n$  is the down-flow received by any  $\Pi_i^t$ . Unless neither **AskH** occurs (nor  $\text{Encrypt}_5$ ) games  $\mathbf{G}_5$  and  $\mathbf{G}_6$  are perfectly indistinguishable:

$$\Pr[S_6 \mid \neg\text{AskH}] = \Pr[S_5 \mid \neg\text{AskH}]. \quad (6)$$

The probability of event **AskH** is upper-bounded in the following lemma. Let us recall that  $q_h$  denotes the number of hash-queries asked by the adversary.

**Lemma 8.**  $\Pr[\text{AskH}] \leq q_h \text{Succ}_{\mathbb{G}}^{\text{tgcdh}_n}(T + (q_s + q_E)n\tau_{\mathbb{G}}).$

In this game, answers to the **Reveal**-queries, and thus to the **Test**-query, are purely random. Then, it is straightforward to see that

$$\Pr[\mathbf{S}_6] = \frac{1}{2}. \quad (7)$$

From Lemma 4 and Equations (6), (7), we get:

$$\left| \Pr[\mathbf{S}_5] - \frac{1}{2} \right| = \left| \Pr[\mathbf{S}_5] - \Pr[\mathbf{S}_6] \right| \leq \Pr[\text{AskH}].$$

Finally, from Equations (1), (2), (3), (4) and (5), and Lemmas 4, 8 we get:

$$\left| \Pr[\mathbf{S}_0] - \frac{1}{2} \right| \leq \Pr[\text{Encrypt}_1] + \frac{q_E^2}{q^2} + q_h \text{Succ}_{\mathbb{G}}^{\text{tgcdh}_n}(T + (q_s + q_E)n\tau_{\mathbb{G}}). \quad (8)$$

### 6.3 Probability of Event $\text{Encrypt}_1$

The security against dictionary attacks is measured by the probability that the event  $\text{Encrypt}_1$  occurs. To evaluate this probability, we define a game wherein the view of the adversary is perfectly independent from the password  $pw$ , in the information theoretical sense. First, let us note that the games  $\mathbf{G}_2$ ,  $\mathbf{G}_3$ ,  $\mathbf{G}_4$  and  $\mathbf{G}_5$  are perfectly indistinguishable, and thus

$$\Pr[\text{Encrypt}_5] = \Pr[\text{Encrypt}_1]. \quad (9)$$

We define an auxiliary game  $\mathbf{G}_6''$  similar to game  $\mathbf{G}_5$  except that we answer differently to a  $\text{Send}(H_i^t, \text{Fl})$ -query when instance  $H_i^t$  is waiting for an up-flow. In this game  $\mathbf{G}_6''$ , we in fact re-define all coefficients used by the random self-reducibility and entirely re-blind line  $L_i$ :

**Rule 1.2:** Whatever appears in  $\Lambda_{\Psi}$  so far, we choose a random exponent  $\nu \in \mathbb{Z}_q^*$ , a full vector  $\theta \in (\mathbb{Z}_q^*)^n$  and compute  $L' = \Psi(L_i, \theta, \nu)$ .

We then update  $\Lambda_{\Psi}$ .

By Lemma 2, the plaintext  $L'$  is indistinguishable from a random plaintext in  $\bar{\mathbb{G}}^{i+1}$  and, therefore, the simulation is completely independent from the password  $pw$ . So we have

$$\Pr[\text{Encrypt}_6''] = q_s/N. \quad (10)$$

Moreover the only difference between games  $\mathbf{G}_6''$  and  $\mathbf{G}_5$  is in the way the **Send**-queries are answered. On input of a line  $L = (a_1, \dots, a_{i-1}, a_i)$ , the **Rule 1.1** generates line  $L' = (a_1^{\nu x}, \dots, a_{i-1}^{\nu x}, a_i^{\nu}, a_i^{\nu x})$  while the **Rule 1.2** generates  $L'' = (g^{r_0}, \dots, g^{r_i})$ . Using the classical hybrid technique we can obtain:

$$\left| \Pr[\text{Encrypt}_6''] - \Pr[\text{Encrypt}_5] \right| \leq q_s \text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(T + (q_s + q_E)n\tau_{\mathbb{G}}). \quad (11)$$

Finally, Equations (9), (10) and (11) lead to

$$\Pr[\text{Encrypt}_1] \leq \frac{q_s}{N} + q_s \text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(T + (q_s + q_E)n\tau_{\mathbb{G}}).$$

Note that  $q_E$  is the size of  $\Lambda_E \cup \Lambda'_E$  and it is thus equal to  $q_e$  plus the number of queries asked by our simulation (at most two per **Send**-query):  $q_E \leq q_e + 2q_s$ . This note, combined with Equation (8), concludes the proof.  $\square$

## 7 Forward-Secrecy

The above proof does not deal with forward-secrecy. Considering forward-secrecy requires to take into account a new kind of query that we call the **Corrupt**-query (any other kinds of queries can still be asked after this one):

- **Corrupt**( $U$ ): This query models the attacks resulting in the password  $pw$  to be revealed.  $\mathcal{A}$  gets back from his query  $pw$  but does not get any internal data of  $U$ .

Then we define a new flavor of freshness, saying that an oracle  $\Pi_i^t$  is **Fresh** (or holds a **Fresh** key  $sk$ ) if the following conditions hold. First, no **Corrupt**-query has been made by the adversary since the beginning of the game. Second,  $\Pi_i^t$  has computed a session key and neither  $\Pi_i^t$  nor its partners have been asked for a **Reveal**-query. The partnering notion is more formally defined in the full version of this paper [8].

This security level means that the adversary does not learn any information about *previously* established session keys when making a **Corrupt**-query. We thus denote by  $\text{Adv}_P^{\text{akefs}}(\mathcal{A})$  the advantage an adversary can get on a fresh key, with the ability to make a **Corrupt**-query.

**Theorem 2.** *Let  $P$  be the EKE protocol,  $\mathbf{SK}$  be the session-key space and Password be a finite dictionary of size  $N$ . Let  $\mathcal{A}$  be an adversary against the AKE security of  $P$  within a time bound  $T$ , after  $q_s$  interactions with the parties,  $q_h$  hash-queries, and  $q_e$  encryption/decryption queries. Then, there exists  $k \leq n$  such that:*

$$\text{Adv}_P^{\text{akefs}}(\mathcal{A}) \leq \frac{2q_s}{N} + 2q_s \text{Adv}_{\mathbb{G}}^{\text{mddh}_n}(T') + 2n^2 q_s^n q_h \text{Succ}_{\mathbb{G}}^{\text{tgcdh}_k}(T') + \frac{2Q^2}{q^2}.$$

where  $T' \leq T + Qn\tau_{\mathbb{G}}$ ,  $Q = 5q_s + q_e$  and  $\tau_{\mathbb{G}}$  is the time of computation required for an exponentiation in  $\mathbb{G}$ . (Recall that  $q$  is the order of  $\mathbb{G}$ ).

*Proof.* The proof of this theorem is given in the full version of this paper [8].

## 8 Mutual Authentication

The well-known approach for turning an Authenticated Key Exchange (AKE) protocol into a protocol that provides mutual authentication (MA) is to use the

shared session key to construct a simple “authenticator” for the other parties. In [10], we already described the transformation, and justified its security in the random-oracle model. The first analysis has been done before in the two-party case in [2].

## 9 Conclusion

This paper provides the first formal treatment of the authenticated group Diffie-Hellman key exchange problem that encompasses dictionary attacks. Addressed in this paper are two security goals of the group Diffie-Hellman key exchange: the authenticated key exchange and the mutual authentication. For each we present a definition, a protocol and a security proof in both the random oracle model and the ideal-cipher model that the protocol meets its goals. Furthermore, we consider forward-secrecy, even if the reduction is not very efficient. Reducing the ideal-cipher model assumption and improving the reduction for the forward-secrecy are still open problems.

## Acknowledgments

The authors thank the anonymous referees for their useful comments. The second author was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Information and Computing Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This document is report LBNL-49479. Disclaimer available at <http://www-library.lbl.gov/disclaimer>.

## References

1. N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. *Computer Communications*, 23(18):1627–1637, 2000.
2. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In B. Preneel, editor, *Proc. of Eurocrypt '00*, LNCS 1807, pages 139–155. Springer-Verlag, 2000.
3. S. M. Bellovin and M. Merrit. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of the Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.
4. J. Black and P. Rogaway. Ciphers with Arbitrary Finite Domains. In *Proc. of the RSA Cryptographer's Track (RSA CT '02)*, LNCS 2271, pages 114–130. Springer-Verlag, 2002.
5. Bluetooth. Specification of the Bluetooth System, December 1999. Available at <http://www.bluetooth.com/developer/specification/specification.asp>.
6. D. Boneh. The Decision Diffie-Hellman Problem. In *Third Algorithmic Number Theory Symposium*, LNCS 1423, pages 48–63. Springer-Verlag, 1998.
7. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In B. Preneel, editor, *Proc. of Eurocrypt '01*, LNCS 1807, pages 156–171. Springer-Verlag, 2000.

8. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In Y. Zheng, editor, *Proc. of Asiacrypt '2002*. Springer, December 2002. Full Version – <http://www.di.ens.fr/users/pointche>.
9. E. Bresson, O. Chevassut, and D. Pointcheval. The Group Diffie-Hellman Problems. In H. Heys and K. Nyberg, editors, *Proc. of SAC '2002*, LNCS. Springer-Verlag, August 2002.
10. E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proc. of 8th ACM Conference on Computer and Communications Security*, pages 255–264, November 2001.
11. W. Diffie and M. Hellman. New Directions In Cryptography. In *IEEE Transactions on Information Theory*, volume IT-22(6), pages 644–654, November 1976.
12. O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. In J. Kilian, editor, *Proc. of Crypto '01*, LNCS 2139, pages 408–432. Springer-Verlag, August 2001.
13. M. Jakobsson and S. Wetzel. Security Weaknesses in Bluetooth. In *Proc. of the RSA Cryptographer's Track (RSA CT '01)*, LNCS 2020, pages 176–191. RSA Data Security, Springer-Verlag, 2001.
14. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords. In *Proc. of Eurocrypt '01*, LNCS 2045, pages 475–494. Springer-Verlag, May 2001.
15. P. MacKenzie. More Efficient Password Authenticated Key Exchange. In D. Naccache, editor, *RSA Conference '01*, LNCS 2020, pages 361–377. Springer-Verlag, 2001.
16. M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. In *Proc. of 38th FOCS*, pages 458–467. IEEE, 1997.
17. NIST. AES, December 2000. Available at <http://www.nist.gov/aes>.
18. K. Obraczka, G. Tsudik, and K. Viswanath. Pushing the Limits of Multicast in Ad Hoc Networks. In *International Conference on Distributed Computing Systems*, April 2001.
19. C. E. Perkins. *Ad Hoc Networking*. Addison Wesley, 2001.
20. V. Shoup. OAEP Reconsidered. In J. Kilian, editor, *Proc. of Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, 2001.
21. L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6), 1999.

# Secure Channels Based on Authenticated Encryption Schemes: A Simple Characterization

Chanathip Namprempre

Department of Computer Science, Thammasat University, 41-42 km. Paholyothin  
Road, Khong Luang, Rangsit, Pathum Thani, Thailand 12121,  
`meaw@alum.mit.edu`, `www-cse.ucsd.edu/users/cnamprem`

**Abstract.** We consider communication sessions in which a pair of parties begin by running an authenticated key-exchange protocol to obtain a shared session key, and then secure successive data transmissions between them via an authenticated encryption scheme based on the session key. We show that such a communication session meets the notion of a secure channel protocol proposed by Canetti and Krawczyk [9] if and only if the underlying authenticated encryption scheme meets two new, simple definitions of security that we introduce, and the key-exchange protocol is secure. In other words, we reduce the secure channel requirements of Canetti and Krawczyk to easier to use, stand-alone security requirements on the underlying authenticated encryption scheme. In addition, we relate the two new notions to existing security notions for authenticated encryption schemes.

## 1 Introduction

We consider communication sessions in which a pair of parties begin by running an authenticated *key-exchange* (KE) protocol to obtain a shared *session key*, and then secure successive data transmissions between them via an *authenticated encryption scheme*, a shared-key-based encryption scheme whose goal is to provide *both* privacy *and* authenticity, based on the session key. Many popular Internet protocols follow this structure [1, 5, 11, 23]. One reason is that it minimizes computationally intensive public-key cryptography by using more efficient symmetric-key cryptography for the bulk of the communication.

At Eurocrypt 2001, Canetti and Krawczyk presented security definitions for protocols of this form [9]. They refer to such protocols as *network channel protocols* (or *channel protocols* for short). In their work, they derive a realistic adversarial model from [2] and formulate security definitions using a mixture of both *simulation-based* and *indistinguishability-based* approaches. The former allows them to realistically and naturally capture the security properties of channel protocols and the settings in which the protocols are deployed. The latter allows them to prove security of the protocols with relative ease. The result is the notion of *secure channels*, a notion that captures the desired security properties of the

communication channels themselves, rather than those of the components used in constructing them, namely the underlying authenticated encryption schemes.

In contrast, most existing work has traditionally focused on security properties of encryption schemes. Examples include indistinguishability notions for asymmetric encryption schemes pioneered in [17] and adapted to symmetric-key settings in [3], non-malleability notions defined in [13,3] and refined in [8], and integrity notions defined in [19,5,20]. Due to the simplicity and ease of use of these definitions, this approach has proved fruitful and has become the standard way to prove security of encryption schemes.

Our work uses this traditional approach to investigate security properties of the authenticated encryption schemes underlying channel protocols. In particular, our goal is to address the following question. Suppose one takes a “secure” KE protocol and combines it with an authenticated encryption scheme as described above to obtain a channel protocol. What are the necessary and sufficient conditions on the underlying authenticated encryption scheme for the resulting channel protocol to be a secure channel per [9]? The answer to this question will allow us to analyze security of channel protocols in a modular fashion: first consider the underlying KE protocol and the underlying authenticated encryption scheme separately, then determine whether the former is “secure” and whether the latter meets the necessary and sufficient conditions. If both are affirmative, then the channel protocol in question is a secure channel. Not only does this approach simplify protocol analysis, but the necessary and sufficient conditions also help distill exactly the security properties of authenticated encryption schemes that are needed to obtain secure channels. This understanding can help guide cryptographers in designing future schemes for building secure channels.

Krawczyk has already made some progress in this direction in [20]: he provides a necessary condition for a class of authenticated encryption schemes, namely those constructed via the “Authenticate-then-Encrypt” method, to yield a secure channel, assuming that the underlying KE protocol is “secure.” Our goal is to provide *both* necessary *and* sufficient conditions that are easy-to-use and can be applied to *any* authenticated encryption schemes, as opposed to schemes of a certain form. To this end, we use the traditional approach of defining security since it yields definitions that are simple and relatively easy to use.

SECURITY MODEL OF CANETTI AND KRAWCZYK. In [9], Canetti and Krawczyk use the adversarial model of [2]: an adversary is in control of all message delivery and the execution of the protocol. In particular, once the setup phase of the protocol is completed, all parties in the system simply wait for *activations* from the adversary. Possible activations include sending messages, receiving messages, and establishing a session. Messages are delivered solely by the adversary under

<sup>1</sup> Under this paradigm, a message authentication scheme and an encryption scheme are composed to obtain an authenticated encryption scheme as follows. To encrypt a message  $M$ , first compute its MAC via a message authentication scheme and encrypt the concatenation of  $M$  and the MAC to obtain the ciphertext to be transmitted. Decryption works in a natural way.



either of the following models: the Authenticated-links Model (AM) and the Unauthenticated-links Model (UM). Both models allow the adversary to drop messages and to deliver them out of order. In the former, an adversary cannot inject messages and must deliver messages without modifications. In the latter, it can inject fabricated messages and modify messages before delivering them. Section 2.1 describes the security model of [9] in more detail.

Canetti and Krawczyk also present a security definition for KE protocols based on the approach of [6] in this adversarial model. Intuitively, they consider a KE protocol to be secure if, when the two parties involved in the exchange complete the protocol, (1) they arrive at the same session key, and (2) it is hard for an adversary to distinguish the session key from a random value chosen from the distribution of keys generated by the protocol.

**SECURE CHANNELS.** Canetti and Krawczyk define a secure channel as a channel protocol that is both a *secure (network) authentication* protocol and a *secure (network) encryption* protocol. The definition of the former uses a simulation-based approach: a protocol secure in this sense must *emulate* ideal message transmissions where the notion of emulation amounts to computational indistinguishability of protocol outputs. To this end, [9] defines a *session-based message transmission* (SMT) protocol, a protocol that does nothing more than its name suggests. For example, to establish a session, a party simply records in its output that a session has been established. To send a message, a party simply puts the message in the message buffer and records in its output that the message has been sent.

The definition of secure encryption protocols applies an indistinguishability-based approach similar to the “find-then-guess” game in [3] (which in turn is an adaptation of *semantic security* of [17] into the symmetric setting) in this adversarial model. Specifically, the protocol is run in the UM against an adversary which, at some point during the run, chooses a session it wishes to break. The rest of the run closely follows the standard find-then-guess game with a few important exceptions. See Section 2.2 for details.

**CAPTURING THE ESSENCE OF SECURE CHANNELS.** Following [9], we define a *transform* to specify how the channel protocols considered in this paper are generated: given a KE protocol  $\pi$  and an authenticated encryption scheme  $\mathcal{AE}$ , we associate with them a channel protocol  $\text{NC} = \text{NetAE}(\pi, \mathcal{AE})$  obtained by applying the transform to  $\pi$  and  $\mathcal{AE}$ . This transform is defined in Section 2.3. We focus on protocols constructed via this transform. Our goal is to find simple necessary and sufficient conditions on the underlying authenticated encryption scheme such that the protocol is a secure channel, assuming that the KE protocol is secure. We define two simple notions: SINT-PTXT and IND-CCVA. The former (resp. the latter) is a necessary and sufficient condition on the underlying authenticated encryption scheme such that the channel protocol is a secure authentication (resp. encryption) protocol. In effect, this reduces the secure channel requirements of Canetti and Krawczyk to easier to use, stand-alone security requirements on the underlying authenticated encryption scheme.

We define the two notions using the traditional approach: we give an adversary access to certain oracles, run it in an experiment, and then measure the probability that it succeeds. Section 3 describes these notions in detail. Precise statements of our main results are presented in Section 4 along with the proof ideas.

**TECHNICAL ISSUE.** The notion of secure authentication protocols captures reasonable authenticity guarantees such as resistance against replay attacks and forgeries. Therefore, to determine if a channel protocol provides authenticity when these attacks are of concern, one needs simply determine whether the protocol is a secure authentication protocol. However, due to a technical issue arisen from the notion of secure encryption protocol per [9], the same cannot be said regarding privacy. In particular, there exists a channel protocol that clearly does not provide semantic security [17] (i.e., partial information about transmitted messages may be leaked) and yet *is* provably a secure encryption protocol. Arguably, however, this technical issue does not arise in many practical protocols, including the popular SSH, SSL, and TLS. Consequently, the notion of secure encryption protocol can still be applied to these protocols to obtain meaningful results regarding their privacy guarantees. Section 5 discusses this issue in more detail.

**FUTURE WORK.** Canetti and Krawczyk have recently proposed an alternative notion for secure channels that implies their secure channel notion of [9]. This new notion is called *universally composable secure channels* [10]. It provides strong composability guarantees, which means that its security guarantees hold even if the channel protocol is used in combination with other protocols. Thus, a natural research direction is to determine whether we can use the same approach taken here to derive simple necessary and sufficient conditions for an authenticated encryption scheme to yield a universally composable secure channel.

## 2 Definitions

### 2.1 Preliminaries

Since the authenticated encryption schemes considered in [9] have stateful decryption algorithms, we modify the standard syntax of symmetric authenticated encryption schemes, which assumes that decryption algorithms are stateless [3], to allow for stateful decryption algorithms. We also explicitly specify the syntax of a message-driven protocol based on [2,9] and restate the security model of [9] in more detail here.

**SYNTAX OF (SYMMETRIC) AUTHENTICATED ENCRYPTION SCHEMES.** A (symmetric) authenticated encryption scheme  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  consists of three algorithms. The randomized *key generation* algorithm  $\mathcal{K}$  takes as input a security parameter  $k \in \mathbb{N}$  and returns a key  $K$ ; we write  $K \xleftarrow{R} \mathcal{K}(k)$ . The *encryption* algorithm  $\mathcal{E}$  could be randomized or stateful. It takes the key  $K$  and a *plaintext*  $M$  to return a *ciphertext*  $C$ ; we write  $C \xleftarrow{R} \mathcal{E}_K(M)$ . The *decryption* algorithm

$\mathcal{D}$  could be deterministic, and it could be either stateless or stateful. It takes the key  $K$  and a string  $C$  to return either the corresponding plaintext  $M$  or the symbol  $\perp$ ; we write  $x \leftarrow \mathcal{D}_K(C)$  where  $x \in \{0, 1\}^* \cup \{\perp\}$ . Above, a randomized algorithm flips coins anew on each invocation, and a stateful algorithm uses and then updates a state that is maintained across invocations.

Since the decryption algorithm is allowed to be stateful here, the usual correctness condition, which requires that  $\mathcal{D}_K(\mathcal{E}_K(M)) = M$  for all  $M$  in the message space, is replaced with a less stringent condition requiring only that decryption succeed when the encryption and decryption processes are in synchrony. More precisely, the following must be true for any key  $K$  and plaintexts  $M_1, M_2, \dots$ . Suppose that both  $\mathcal{E}_K$  and  $\mathcal{D}_K$  are in their initial states. For  $i = 1, 2, \dots$ , let  $C_i = \mathcal{E}_K(M_i)$  and let  $M'_i = \mathcal{D}_K(C_i)$ . It must be that  $M_i = M'_i$  for all  $i$ . Notice that this imposes no correctness requirement when ciphertexts are decrypted out of order. It is up to an individual scheme to decide how to handle ciphertexts that are decrypted out of order. For example, it can reject all such ciphertexts or accept only the ones that decrypt to certain seen messages. We stress that since this requirement is a part of the *syntax* of encryption schemes, it is liberal by design (messages that arrive out of order can have arbitrary decryptions under this requirement!)<sup>2</sup> The goal here is to ensure that as many encryption schemes as possible can be analyzed under the security notions of interest.

**SYNTAX OF MESSAGE-DRIVEN PROTOCOLS.** A message-driven protocol  $\text{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$  consists of three algorithms, four positive integer parameters, and a list of *activations* that can be invoked on a party along with instructions on how the party should handle them. Let  $k \in \mathbb{N}$  be the security parameter. The parameter  $n$  specifies the upper bound of the number of parties in the system. The randomized *input generation* algorithm  $\mathcal{IG}$  takes as inputs  $k$  and an  $x$ -bit string and returns  $n$  strings  $(x_1, \dots, x_n)$ . The randomized *bootstrapping* algorithm<sup>3</sup>  $\mathcal{B}$  takes as inputs  $k$  and an  $l$ -bit string and returns  $n + 1$  strings  $(I_0, \dots, I_n)$ . For each party  $P_i$ , the possibly randomized *initialization* algorithm  $\mathcal{I}$  takes as inputs  $I_0, I_i, x_i$ , and an  $r$ -bit string. Executing the initialization algorithm may cause the party to update its *internal state*, to generate outputs to be appended to its *local output*, and/or to produce messages to be sent to other parties.

**MESSAGE-DRIVEN PROTOCOL EXECUTION** [9]. Let  $k \in \mathbb{N}$  be the security parameter. A protocol  $\text{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$  is executed against an adversary as follows. First, random coins for  $\mathcal{IG}, \mathcal{B}$ , and  $\mathcal{I}$  are generated, and  $\mathcal{IG}$  and  $\mathcal{B}$  are executed. Then, each party  $P_i$  executes the initialization algorithm  $\mathcal{I}$  giving it appropriate inputs as described above. When the initialization algo-

<sup>2</sup> Recall that *syntax* and *security notion* are two separate concepts. Apparently “insecure” schemes such as one that allows arbitrary decryptions for messages that arrive out-of-order are in fact legitimate encryption schemes, i.e. they follow the syntax defined here. However, they are not secure under integrity notions, for instance.

<sup>3</sup> Also known as an initialization function in [29]. We drop their terminology here to avoid confusion with the initialization algorithm.

rithm completes, the party waits for incoming activations. Finally, the adversary is run using  $k, I_0$ , and as many random coins as it needs. The adversary takes over and activates any parties it wishes to at this point.

Upon receiving an activation, a party executes the corresponding algorithm as specified in **activation list**. Again, the result of the execution may be internal state updates, local output generation, and/or *outgoing messages*. In the last case, the party appends the message in the *message buffer*  $\mathcal{M}$  along with its source, destination, and, in the case of a session-based protocol, the associated session. As an example, upon receiving a “send” activation from the adversary, a party finds the algorithm for handling a send activation in its activation list and executes the algorithm. This typically involves encrypting the message, appending the ciphertext (along with its source, destination, and session ID) to  $\mathcal{M}$ , and recording the event (e.g., a record to the effect “sent  $M$  to  $P$  within session  $s$ ”) in the party’s local output.

**PROTOCOL OUTPUT.** The output of a running protocol is the concatenation of the cumulative local outputs of all the parties, together with the output of the adversary. Furthermore, since all actions of the adversary are recorded in the local outputs, they are part of the protocol output.

**SESSION-BASED MESSAGE-DRIVEN PROTOCOLS** [9]. A *session-based message-driven* protocol defines at least two activations: **establish-session** and **expire-session**. They specify how each party can establish a *session* between itself and another. We denote by  $(P, P', s)$  a session defined by the initiating party  $P$ , the responding party  $P'$ , and the session ID  $s$ . The two parties  $P$  and  $P'$  are said to play the *roles* of an **initiator** and a **responder**, respectively. Two identical sessions (i.e., identical session IDs, participating parties, and their respective roles) from the point of view of the initiator and the responder are called *matching sessions*. The defining feature of session-based protocols is that individual sessions are maintained separately from one another even when they are established between the same pair of parties.

**KEY-EXCHANGE PROTOCOLS.** A *key-exchange (KE)* protocol is a session-based message-driven protocol that specifies how two parties can establish a shared *session key* to be used during a session. Upon an **establish-session** activation, a party triggers a sub-protocol to establish a session with another party. This sub-protocol will likely result in further activations such as message sends and receipts. Once the sub-protocol completes, the two parties write on their outputs the resulting session key and mark the entry as “secret.” Note that, although potentially confusing, the term “key-exchange protocol” is commonly used in the literature to refer to this sub-protocol rather than the entire protocol. Upon an **expire-session** activation of a particular session, the party erases the corresponding session key from its output and any internal state it may have (e.g., its memory) and terminate the session. Notice that this means that a session can be unilaterally expired. The goal of this activation is to allow KE protocols to provide *perfect forward secrecy* of sessions, a property that past session keys remain secret even after long-term keys are compromised [18][12].

NETWORK CHANNEL PROTOCOLS. A *network channel* protocol (or a channel protocol for short) is a session-based message-driven protocol with two additional activations: **send** and **incoming**. They specify what a party running the protocol should do to send and to receive a message.

POWER OF AN ADVERSARY. When interacting with parties executing a session-based message-driven protocol, an adversary is allowed to access the contents of each party's local output except those marked as “secret.” It can also perform the following *actions*: **party activation**, **party corruption**, **session-state reveal**, and **session-output reveal**. In addition to these actions, an adversary against a KE protocol can also perform a **session-key reveal** action against a party to obtain a session key. A session is considered *exposed* if it belongs to a corrupted party, has been subjected to a **session-state reveal**, a **session-output reveal**, a **session-key reveal**, or has a matching session that has been exposed.

AUTHENTICATED AND UNAUTHENTICATED LINKS MODELS. In the Authenticated-links Model (AM), the adversary can perform all of the actions mentioned above. Furthermore, all message delivery is performed by  $A$ : to deliver a message in the message buffer  $\mathcal{M}$ , the adversary  $A$  removes it from  $\mathcal{M}$  and activates the receiving party with the message as an incoming message. We emphasize that  $A$  can deliver messages in any arbitrary order and can drop messages from  $\mathcal{M}$  entirely. However, it cannot deliver messages that are not in  $\mathcal{M}$ , and when it does deliver a message, it must do so without any modifications to the message. On the other hand, in the Unauthenticated-links Model (UM), not only can a UM adversary perform all of the actions permitted to an AM adversary, but it can also deliver messages that are not in  $\mathcal{M}$  or modify messages in  $\mathcal{M}$  before delivering them.

NOTATION. We use  $|r|$  to denote the length in bits of a string  $r$ . Let  $k \in \mathbb{N}$  be the security parameter, and let  $U$  be an adversary. Let  $\text{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$  be a session-based message-driven protocol. We follow the notation of [29] for the protocol output. We describe it here in detail for the UM. The AM is done similarly except that the bootstrapping algorithm is ignored and its outputs are omitted. We denote by  $\text{UNADV}_{\pi,U}(k, \vec{x}, \vec{r})$  the output of the UM adversary  $U$  running against parties executing the protocol  $\pi$  with security parameter  $k$ , inputs  $\vec{x} = (x_1, \dots, x_n)$ , and coins  $\vec{r} = r', r'', r_0, \dots, r_n$  where  $|r'| = x, |r''| = l$ , and  $|r_0| = \dots = |r_n| = r$ . We denote by  $\text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_i$  the cumulative output of the party  $P_i$  running the protocol  $\pi$  with security parameter  $k$ , inputs  $\vec{x}$ , and coins  $\vec{r}$  against the UM adversary  $U$ . Then, we let the protocol output  $\text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r}) = \text{UNADV}_{\pi,U}(k, \vec{x}, \vec{r}), \text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_1, \dots, \text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_n$  and let  $\text{UNAUTH}_{\pi,U}(k)$  be the random variable describing  $\text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})$  when  $\vec{r}$  is randomly chosen and  $\vec{x}$  is generated via  $\mathcal{IG}(k, r')$ . We denote by  $\text{UNAUTH}_{\pi,U}$  the ensemble  $\{\text{UNAUTH}_{\pi,U}(k)\}_{k \in \mathbb{N}}$ .

## 2.2 Secure Channels per Canetti and Krawczyk [9]

In [9], Canetti and Krawczyk define a secure channel as a channel protocol that is both a (secure) *authentication protocol* and a (secure) *encryption protocol*.

For authentication protocols, their approach is to first define a protocol considered ideal as a message authentication protocol called the *SMT* protocol. A channel protocol is considered a secure authentication protocol if it emulates the SMT protocol in the UM. Below, we present the concept of protocol emulation, the SMT protocol, and the definition of secure authentication protocols in Definition 1, Construction 2, and Definition 3, respectively.

**Definition 1 (Protocol Emulation [2,9]).** Let  $\pi, \pi'$  be message-driven protocols. We say that  $\pi'$  *emulates*  $\pi$  in the UM if, for any UM adversary  $U$ , there exists an AM adversary  $A$  such that  $\text{AUTH}_{\pi,A}$  and  $\text{UNAUTH}_{\pi',U}$  are computationally indistinguishable. ■

**Construction 2 (SMT Protocol [9]).** The protocol SMT is a session-based message-driven protocol with the following activations: *establish-session*, *expire-session*, *send*, and *incoming*. Upon an *establish-session* activation, a party records the event accordingly in its output. Upon an *expire-session* activation, a party checks that the session exists, marks the session as expired, and records the event accordingly in its output. When a party receives a *send* activation involving a message, a partner, and a session ID, it checks that the session is established and is not expired. If so, it sends the given message to its partner via the specified session. Then, it records the event accordingly in its output. Finally, upon an *incoming* activation, a party checks that the session is established and is not expired. If so, it records the event accordingly in its output. ■

**Definition 3 (Network Authentication Protocol Security [9]).** A protocol is considered to be a *secure authentication protocol* if it emulates the SMT protocol in the UM. ■

In defining secure encryption protocols, [9] adapts the indistinguishability-based approach to a multi-party computation setting. We present their security definition here. In what follows, the activation  $\text{send}^*(P, Q, s, M_b)$  has the same effects as  $\text{send}(P, Q, s, M_b)$  except that the party  $Q$  merely records the fact that a message is sent but not the actual contents of the message, i.e.,  $P$  records the entry “**sent a message to  $Q$  within session  $s$** ”. Similarly, the activation  $\text{incoming}^*(Q, P, s, C, M_b)$  has the same effects as  $\text{incoming}(Q, P, s, C)$  except that, if the decrypted message of  $C$  is equal to  $M_b$ , then  $Q$  merely records the fact that a message is received but not the actual contents of the message  $M_b$ , i.e.,  $Q$  records the entry “**received a message from  $P$  within session  $s$** ”.

Let  $b$  be a bit. In the experiment below, an adversary  $U$  runs in the UM, and its goal is to break one session of its choice by performing an action called *test-session* against the session and then doing what it can to guess the bit  $b$ . Once  $U$  picks a session, say  $(P, Q, s)$ , it outputs a pair of messages, say  $(M_0, M_1)$ . The sender  $P$  is then activated to send  $M_b$ . However, if  $P$  records in its local output at this point that it sends  $M_b$ , then  $U$  can easily win the game by simply looking at  $P$ ’s output. Therefore,  $P$  is activated with  $\text{send}^*(P, Q, s, M_b)$ , rather than a regular *send* activation. The rest of the run continues in the same

way as before except that now the receiving party of the tested session uses  $\text{incoming}^*(Q, P, s, C, M_b)$  to handle incoming messages. The reason for this is the following: if  $Q$  records all decryptions of incoming ciphertexts,  $U$  can easily determine the bit  $b$  by simply taking the challenge ciphertext corresponding to  $M_b$ , handing it to  $Q$  as an incoming ciphertext, then seeing what  $Q$  writes on its output. The activation  $\text{incoming}^*$  prevents this trivial attack.

Unfortunately, the game in its present form allows  $U$  to easily win via another trivial attack. Suppose the tested session is  $(P, Q, s)$ . First,  $U$  picks any message  $M$ , activates  $P$  with a  $\text{send}$  activation to send  $M$  to  $Q$  via  $s$ , and outputs the challenge message pair  $(M, M')$  where  $M \neq M'$ . As a result of the  $\text{send}$  activation,  $P$  encrypts  $M$  to obtain a ciphertext  $C$  and appends  $C$  to the message buffer. Now,  $U$  activates the receiver  $Q$  with the ciphertext  $C$  as an incoming message from  $P$  via session  $s$ . If  $Q$  does not record the decrypted message, then  $C$  corresponds to  $M$ , and thus  $b = 0$ . Otherwise,  $C$  corresponds to  $M'$ , and thus  $b = 1$ . Therefore, to prevent this trivial attack, [9] requires that an adversary never ask for an encryption of a particular message more than once. This requirement can be easily implemented using counters. For example, the encryption algorithm can prepend an internal counter to the input message before encrypting the resulting string to obtain the ciphertext. In fact, the use of this mechanism is common in practical Internet protocols including SSH [23], SSL [15], and TLS [11]. Definition 4 below describes the security of network encryption protocols more precisely.

**Definition 4 (Network Encryption Protocol Security [9]).** Let  $k \in \mathbf{N}$ . Let  $\text{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$  be a channel protocol. Let  $U$  be a UM attacker, and let  $r_U: \mathbf{N} \rightarrow \mathbf{N}$  be the function specifying the upper bound of the running time of  $U$  in terms of  $k$ . Consider the following experiment:

Experiment  $\text{Exp}_{\text{NC}, U}^{\text{ind-ne-}b}(k)$

$r' \xleftarrow{R} \{0, 1\}^x$ ;  $r'' \xleftarrow{R} \{0, 1\}^l$ ;  $r_0 \xleftarrow{R} \{0, 1\}^{r_U(k)}$

$(x_1, \dots, x_n) \leftarrow \mathcal{IG}(k, r')$ ;  $(I_0, \dots, I_n) \leftarrow \mathcal{B}(k, r'')$

For  $i = 1, \dots, n$  do  $r_i \xleftarrow{R} \{0, 1\}^r$ ; start  $P_i$  on  $(I_0, I_i, x_i, r_i)$

Run  $U$  on input  $(k, I_0, r_0)$ , carrying out  $U$ 's actions as specified in  $\text{NC}$

▷ When  $U$  submits  $\text{test-session}(P_i, P_j, s_0)$  and outputs  $(M_0, M_1)$

— Activate  $P_i$  with  $\text{send}^*(P_i, P_j, s_0, M_b)$

▷ Continue carrying out  $U$ 's actions as specified in  $\text{NC}$  *except*

— Whenever  $U$  activates  $P_j$  with  $\text{incoming}(P_j, P_i, s_0, C)$ ,  
Activate  $P_j$  with  $\text{incoming}^*(P_j, P_i, s_0, C, M_b)$  instead

Until  $U$  halts and outputs a bit  $d$

Output  $d$

Above, it is required that  $U$  submit only one  $\text{test-session}$  query and that it not expose the tested session thereafter. Furthermore, for the tested session, we require that  $U$  never invoke  $\text{send}$  activations involving  $M_0$  or  $M_1$  and also never invoke  $\text{send}$  activations involving a particular message more than once. We define the advantage of the adversary via

$$\text{Adv}_{\text{NC}, U}^{\text{ind-ne}}(k) = \Pr[\text{Exp}_{\text{NC}, U}^{\text{ind-ne-}1}(k) = 1] - \Pr[\text{Exp}_{\text{NC}, U}^{\text{ind-ne-}0}(k) = 1] .$$



The channel protocol  $\mathbf{NC}$  is said to be a *secure encryption protocol* in the UM if the function  $\mathbf{Adv}_{\mathbf{NC}, U}^{\text{ind-ne}}(\cdot)$  is negligible for any UM adversary  $U$  whose time-complexity is polynomial in  $k$ . ■

### 2.3 From KE and Authenticated Encryption Schemes to Channel Protocols

In [9], Canetti and Krawczyk use a template by which one can describe how a KE protocol and an authenticated encryption scheme can be used as building blocks for a channel protocol. We define a transform based on this template.

**Construction 5 (Transform [9]).** Let  $\pi = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$  be a KE protocol, and let  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an authenticated encryption scheme. We associate with  $\pi$  and  $\mathcal{AE}$  a channel protocol  $\mathbf{NAE} = \mathbf{NetAE}(\pi, \mathcal{AE}) = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{alist})$  where alist contains the activations in activation list together with the following activations.

1. **establish-session**( $P_i, P_j, s, \text{role}$ ): This triggers a KE-session under  $\pi$  within  $P_i$  with partner  $P_j$ , session ID  $s$ , and  $\text{role} \in \{\text{initiator}, \text{responder}\}$ . If the KE-session completes,  $P_i$  records in its local output the entry “**established session  $s$  with  $P_j$** ” and the generated session key marked as “secret.” Otherwise, no action is taken.
2. **expire-session**( $P_i, P_j, s$ ): If the session  $(P_i, P_j, s)$  exists at  $P_i$ , the party  $P_i$  marks the session as expired and erases the session key. Then,  $P_i$  records in its local output “**expired session  $s$  with  $P_j$** ”. Otherwise, no action is taken.
3. **send**( $P_i, P_j, s, M$ ): The party  $P_i$  checks that the session  $(P_i, P_j, s)$  has been completed and not expired. If so, it computes  $C \xleftarrow{R} \mathcal{E}_K(M)$  using the corresponding session key  $K$ , puts  $(P_i, P_j, s, C)$  in the message buffer  $\mathcal{M}$ , and records “**sent  $M$  to  $P_j$  within session  $s$** ” in the local output. Otherwise, no action is taken.
4. **incoming**( $P_j, P_i, s, C$ ): The party  $P_j$  checks that the session  $(P_i, P_j, s)$  has been completed and not expired. If so, it computes  $M \leftarrow \mathcal{D}_K(C)$  under the corresponding session key  $K$ . If  $M \neq \perp$ , then  $P_j$  records “**received  $M$  from  $P_i$  within session  $s$** ”. Otherwise, no action is taken. ■

## 3 Simple Characterizations of Authenticated Encryption Schemes for Secure Channels

We propose two new security notions for authenticated encryption schemes: SINT-PTXT (for strong integrity of plaintexts) and IND-CCVA (for indistinguishability against chosen-ciphertext attacks with verification). The goal is to capture the necessary and sufficient properties of the authenticated encryption scheme such that, once the transform per Construction 5 is applied to the scheme and a KE protocol, the resulting channel protocol is a secure channel, assuming that the KE protocol “securely implements” the key generation algorithm of the



authenticated encryption scheme. We postpone a precise definition of the term in quotes to Section 4. In what follows, we use  $x \stackrel{R}{\leftarrow} f(y)$  to denote the process of running a possibly randomized algorithm  $f$  on an input  $y$  and assigning the result to  $x$ . If  $A$  is a program,  $A \leftarrow x$  means “return  $x$  to  $A$ .” The *time-complexity* referred to in our definitions is the worst case total execution time of the entire experiment, plus the size of the code of the adversary, in some fixed RAM model of computation. Also, oracles corresponding to stateful algorithms maintain their states across invocations.

First, we capture the notion of a secure authentication protocol with SINT-PTXT. Recall that a protocol is considered a secure authentication protocol if it emulates the SMT protocol in the UM where SMT is an ideal session-based message transmission protocol. Under the SMT protocol in the AM, when a party sends a message  $M$  to another party, the message  $M$  is simply put on the buffer. Since the adversary is operating in the AM, it can drop messages but cannot modify or inject messages. Therefore, a secure authentication protocol must ensure that each sent message is received *at most once* (i.e., replay attacks are unsuccessful), and that its contents are left intact.

We define the SINT-PTXT notion in Definition 6. An adversary is given access to an encryption oracle and a decryption oracle. This captures its ability to obtain encryption and decryption of messages and ciphertexts of its choice. We use a multiset, denoted  $T$  below, to keep track of messages that have been sent but not yet received. Whenever a message is received, it is removed from the multiset. If an adversary is able to submit a query to the decryption oracle that results in a message that is not in the multiset  $T$ , i.e., the message is not one of those waiting to be received, then it wins.

**Definition 6 (SINT-PTXT).** Let  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an authenticated encryption scheme. Let  $k \in \mathbb{N}$ . Let  $A$  be an adversary with access to two oracles. Consider the following experiment.

Experiment  $\mathbf{Exp}_{\mathcal{AE}, A}^{\text{sint-ptxt}}(k)$   
 $K \stackrel{R}{\leftarrow} \mathcal{K}(k); T \leftarrow \emptyset \quad // \text{ } T \text{ is a multiset}$   
 Run  $A^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)}(k)$   
   Reply to  $\mathcal{E}_K(M)$  as follows:  
      $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M); T \leftarrow T \cup \{M\}; A \leftarrow C$   
   Reply to  $\mathcal{D}_K(C)$  as follows:  
      $M \leftarrow \mathcal{D}_K(C)$   
     If  $M = \perp$  Then  $A \leftarrow M$   
     Else If  $M \in T$  Then  $T \leftarrow T - \{M\}; A \leftarrow M$   
     Else return 1  
 Until  $A$  halts  
 Return 0

We define the *advantage* of the adversary via

$$\mathbf{Adv}_{\mathcal{AE}, A}^{\text{sint-ptxt}}(k) = \Pr[\mathbf{Exp}_{\mathcal{AE}, A}^{\text{sint-ptxt}}(k) = 1] .$$

The scheme  $\mathcal{AE}$  is said to be *SINT-PTXT secure* if the function  $\mathbf{Adv}_{\mathcal{AE},A}^{\text{ sint-ptxt}}(\cdot)$  is negligible for any adversary  $A$  whose time-complexity is polynomial in  $k$ . ■

Now, we capture the notion of a secure encryption protocol. To capture an adversary's ability to obtain encryption and decryption of messages and ciphertexts of its choice, we give it access to an encryption oracle  $\mathcal{E}_K(\cdot)$  and a decryption oracle  $\mathcal{D}_K(\cdot)$ . The definition follows that of [9] closely and straightforwardly. Let  $b \in \{0, 1\}$ . Recall that, in the definition of secure encryption protocol per [9], once the adversary outputs a challenge message pair  $(M_0, M_1)$ , the receiver of the tested session does not record the decrypted message if it is equal to the secret message  $M_b$ . Therefore, we capture this through an oracle denoted by  $\mathcal{D}_K(\cdot, M_b)$ . This oracle is the same as the standard decryption oracle  $\mathcal{D}_K(\cdot)$  except the following. If a given ciphertext decrypts to  $M_b$ , then the oracle  $\mathcal{D}_K(\cdot, M_b)$  returns a special symbol  $\pm$ . Otherwise, it returns the decrypted message. Additionally, since an adversary in the definition per [9] cannot obtain encryptions of a particular message more than once, we also impose the same restriction on the adversary in our experiment.

**Definition 7 (IND-CCVA).** Let  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an authenticated encryption scheme. Let  $b \in \{0, 1\}$  and  $k \in \mathbb{N}$ . Let  $A$  be an adversary that has access to three oracles. Consider the following experiment.

Experiment  $\mathbf{Exp}_{\mathcal{AE},A}^{\text{ ind-ccva-}b}(k)$

$$\begin{aligned} K &\xleftarrow{R} \mathcal{K}(k) \\ (M_0, M_1, st) &\leftarrow A^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)}(k, \text{find}) \\ C &\xleftarrow{R} \mathcal{E}_K(M_b) \\ d &\leftarrow A^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot, M_b)}(k, \text{guess}, C, st) \\ &\text{Return } d \end{aligned}$$

The computation  $\mathcal{E}_K(M_b)$  above is a call to the encryption oracle. Also, the oracle  $\mathcal{D}_K(\cdot, M_b)$  shares states with (i.e., is initialized with the current states of)  $\mathcal{D}_K(\cdot)$  if any. Furthermore, we require that  $A$  never query  $\mathcal{E}_K(\cdot)$  on  $M_0$  or  $M_1$  and also never query  $\mathcal{E}_K(\cdot)$  on a particular message more than once. We define the *advantage* of the adversary via

$$\mathbf{Adv}_{\mathcal{AE},A}^{\text{ ind-ccva}}(k) = \Pr[\mathbf{Exp}_{\mathcal{AE},A}^{\text{ ind-ccva-1}}(k) = 1] - \Pr[\mathbf{Exp}_{\mathcal{AE},A}^{\text{ ind-ccva-0}}(k) = 1] .$$

The scheme  $\mathcal{AE}$  is said to be *IND-CCVA secure* if the function  $\mathbf{Adv}_{\mathcal{AE},A}^{\text{ ind-ccva}}(\cdot)$  is negligible for any adversary  $A$  whose time-complexity is polynomial in  $k$ . ■

## 4 SINT-PTXT and IND-CCVA are Necessary and Sufficient

Our results use Definition 8 below. It describes how a key generation algorithm of an authenticated encryption scheme should relate to a KE protocol of a channel protocol based on the authenticated encryption scheme. In particular, the KE protocol should “implement” the key generation algorithm, meaning that two

parties that have completed the KE protocol with each other should end up with the same key which in turn should be drawn from the distribution generated by the key generation algorithm. The definition, which is adapted from [9], captures this property more precisely via the following game. Let  $k \in \mathbb{N}$  be the security parameter. Let  $\Pi$  be a session-based message-driven protocol that includes a KE protocol  $\pi$  as a sub-protocol, and let  $U$  be a UM adversary running against  $\Pi$ . The adversary  $U$  can carry out actions specified in  $\Pi$  plus one additional activation, namely a **test-session-key** query, against at most one unexpired and unexposed session  $s$  whose KE portion is completed. From this point on,  $U$  is not allowed to expose the tested session. Once  $U$  perform a **test-session-key** query, a bit  $b$  is chosen at random. If  $b = 0$ , then  $U$  receives the session key for  $s$ . Otherwise, it receives a value  $r \xleftarrow{R} \mathcal{K}(k)$ . The adversary wins if it correctly guesses the bit  $b$ .

**Definition 8 (Securely Implementing a Key Generation Algorithm via a Key Exchange Protocol).** Let  $k \in \mathbb{N}$  be the security parameter. A KE protocol  $\pi$  is said to *securely implement* a key generation algorithm  $\mathcal{K}$  in the UM during the run of a protocol if, for any adversary  $U$  in the UM,

- When an uncorrupted party completes  $\pi$  with another uncorrupted party, they both arrive at the same session key, AND
- $U$  wins the game above with probability no more than  $1/2$  plus a negligible function of  $k$ . ■

We present our main results here. They state that, respectively, SINT-PTXT and IND-CCVA are necessary and sufficient for the notions of network authentication and network encryption of Canetti and Krawczyk [9]. We present the theorems and their proof ideas below. The full proofs in detail are in the full version of this paper [21]. For brevity, we write  $X \stackrel{s}{\approx} Y$  when the ensembles  $X$  and  $Y$  are *statistically indistinguishable*. Note that statistical indistinguishability implies computational indistinguishability.

**Theorem 9 (Given a Secure KE, SINT-PTXT  $\Leftrightarrow$  Secure Authentication Protocol).** Let  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an authenticated encryption scheme, and let  $\pi$  be a KE protocol. Let  $\text{NAE} = \text{NetAE}(\pi, \mathcal{AE})$  be the associated channel protocol as per Construction 5. Suppose that  $\pi$  securely implements  $\mathcal{K}$  in the UM during the run of NAE. Then,  $\mathcal{AE}$  is SINT-PTXT secure if and only if NAE is a secure authentication protocol. ■

We sketch the proof for each direction of the “if and only if,” assuming throughout that  $\pi$  securely implements  $\mathcal{K}$ . For the “if” direction, we show that if  $\mathcal{AE}$  is SINT-PTXT, then given any UM adversary  $U$  against NAE, we can construct an AM adversary  $A$  against SMT such that  $\text{AUTH}_{\text{SMT}, A} \stackrel{s}{\approx} \text{UNAUTH}_{\text{NAE}, U}$ . The crux of this proof is essentially the same as that of Theorem 12 of [9], and thus, we do not discuss it further.

For the “only if” direction, we show that, given any sint-ptxt adversary  $F$  against  $\mathcal{AE}$ , we can construct a UM adversary  $U$  against NAE such that, for any AM adversary  $A$  against SMT,  $\text{AUTH}_{\text{SMT}, A} \not\stackrel{s}{\approx} \text{UNAUTH}_{\text{NAE}, U}$  as fol-

lows. The adversary  $U$  starts two parties  $P_1$  and  $P_2$ . Then, it activates  $P_1$  with  $\text{establish-session}(P_1, P_2, s, \text{initiator})$  and runs  $F$ . Whenever  $F$  submits an encryption query  $\mathcal{E}_K(M)$ , the adversary  $U$  activates the party  $P_1$  with  $\text{send}(P_1, P_2, M, s)$ . Similarly, whenever  $F$  submits a decryption query  $\mathcal{D}_K(C)$ , the adversary  $U$  activates the party  $P_2$  with  $\text{incoming}(P_2, P_1, C, s)$ . Recall that a successful sint-ptxt adversary  $F$  can essentially replay a message or forge a ciphertext the decrypts to a previously-unseen message. Since such actions are not allowed in the AM, there can be no AM adversaries that can generate the global output that is statistically indistinguishable from that generated by  $U$ .

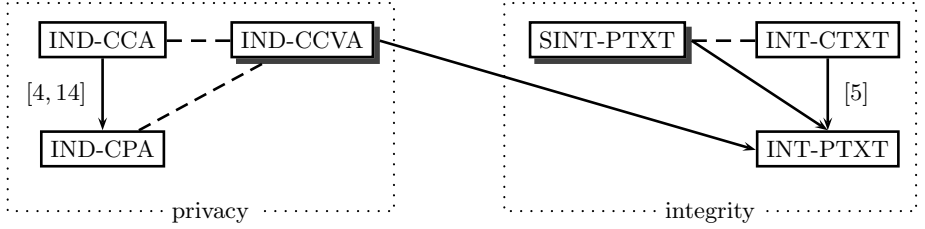
**Theorem 10 (Given a Secure KE,  $\text{IND-CCVA} \Leftrightarrow \text{Secure Encryption Protocol}$ ).** *Let  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an authenticated encryption scheme, and let  $\pi$  be a KE protocol. Let  $\text{NAE} = \text{NetAE}(\pi, \mathcal{AE})$  be the associated channel protocol as per Construction 7. Suppose that  $\pi$  securely implements  $\mathcal{K}$  in the UM during the run of NAE. Then,  $\mathcal{AE}$  is IND-CCVA secure if and only if NAE is a secure encryption protocol. ■*

We sketch the proof for each direction of the “if and only if,” assuming throughout that  $\pi$  securely implements  $\mathcal{K}$ . For the “if” direction, we show that, given any ind-ne adversary  $U$  against NAE, we can construct an ind-ccva adversary  $A$  against  $\mathcal{AE}$  such that  $A$ ’s success probability is no less than that of  $U$  divided by the total number of sessions established by  $U$  over its run. The adversary  $A$  simply simulates  $U$  as in the experiment  $\text{Exp}_{\text{NAE}, U}^{\text{ind-ne-}b}(k)$  (where  $b$  is a bit) with one exception: during the find phase,  $A$  chooses a session at random and uses its oracles to encrypt and decrypt messages in this session. If  $U$  submits a test-session query on the chosen session and outputs a pair of test messages,  $A$  does too. (Otherwise,  $A$  aborts.) Then,  $A$  enters its guess phase and continues the simulation exactly as before. It halts and outputs what  $U$  outputs. Since  $\pi$  securely implements  $\mathcal{K}$ , the adversary  $A$  correctly simulates  $U$ . Thus, it succeeds if  $U$  does.

For the “only if” direction, we show that, given any ind-ccva adversary  $A$  against  $\mathcal{AE}$ , we can construct an ind-ne adversary  $U$  against NAE such that  $U$ ’s success probability is no less than that of  $A$  using a similar technique as before:  $U$  establishes a session between two parties, then runs  $A$ , answering its encryption and decryption queries by making  $\text{send}$  and  $\text{incoming}$  activations respectively for the session. Finally,  $U$  halts and outputs what  $A$  outputs. Since  $\pi$  securely implements  $\mathcal{K}$ , the adversary  $U$  correctly simulates  $A$ . Thus, it succeeds if  $A$  does.

## 5 Understanding Secure Channels through SINT-PTXT and IND-CCVA

We explore the new notions by taking the standard approach of relating them to familiar notions. Since the two notions are necessary and sufficient for secure channels, the knowledge we gain from this exercise is applicable to secure channels as well. In our comparisons, we use the following terminology. Suppose



**Fig. 1. Relations among notions of symmetric encryption:** An arrow from a notion  $X$  to a notion  $Y$  denotes that  $X$  is *strictly stronger* than  $Y$ . A dashed line between a notion  $X$  and a notion  $Y$  denotes that the two notions are *incomparable*. The relations established in other papers are annotated with the corresponding citations. For simplicity, only interesting relations are shown here. We emphasize that the existing notions in this figure (those in unshaded frames) are variants of the standard notions in the literature. In particular, the oracles here maintain states across invocations.

$X$  and  $Y$  are security notions. We say that  $X$  *implies*  $Y$  if any scheme secure under  $X$  is secure under  $Y$ . We say that  $X$  does *not* imply  $Y$  if there exists an encryption scheme that is secure under  $X$  but is insecure under  $Y$ . We say that  $A$  is *equivalent* to  $B$  if  $A$  implies  $B$  and vice versa. We say that  $X$  is *strictly stronger* than  $Y$  if  $X$  implies  $Y$  but  $Y$  does not imply  $X$ . Finally, we say that  $X$  and  $Y$  are *incomparable* if  $X$  does not imply  $Y$  and if  $Y$  does not imply  $X$ .

In this section, we discuss relations among notions of symmetric encryption as summarized in Figure 1. Our strategy for showing that  $X$  implies  $Y$  is the standard reduction approach: given an adversary that successfully breaks the scheme under the notion  $Y$ , construct an adversary that successfully breaks the scheme under the notion  $X$ . To show that  $X$  does not imply  $Y$ , we start with a scheme secure under  $X$ , then modify it to obtain a scheme that remains secure under  $X$  but is insecure under  $Y$ .

The standard privacy notions we consider here are indistinguishability under chosen-plaintext and adaptive chosen-ciphertext attacks (IND-CPA and IND-CCA). The original definitions of these notions were in the asymmetric setting [17, 16, 13, 22] but can be “lifted” to the symmetric setting using the encryption oracle based template of [3]. We use the “find-then-guess” definitions per [3] throughout our discussions here. In particular, for both notions, an adversary  $A$  plays a game in which it is to “find” a pair of challenge messages  $(M_0, M_1)$ , obtain the ciphertext corresponding to the encryption of one of the challenge messages, and then “guess” a bit indicating to which challenge message the ciphertext corresponds. For IND-CPA,  $A$  is given access to an encryption oracle throughout the game. For IND-CCA,  $A$  is given access to both an encryption oracle and a decryption oracle throughout the game. (This notion is also known as IND-CCA2 [4].)

The integrity notions considered here are integrity of plaintexts [5] and integrity of ciphertexts [7, 19, 5]. An adversary attacking a scheme under these notions is given access to two oracles: a standard encryption oracle and a verifi-

cation oracle—an oracle that returns a bit indicating whether the given ciphertext is valid, i.e., whether it decrypts to  $\perp$ . An adversary succeeds in breaking a scheme under the INT-PTXT notion if it can forge a ciphertext that decrypts to a “new” message, i.e., a message that has not been submitted to the encryption oracle before. Similarly, it succeeds in breaking a scheme under the INT-CTXT notion if it can forge a “new” and valid ciphertext, i.e., a valid ciphertext that has not been returned by the encryption oracle.

Strictly speaking, the original definitions of the existing security notions considered here, namely IND-CPA, IND-CCA, INT-PTXT and INT-CTXT, do not explicitly deal with encryption schemes with stateful decryption algorithms. Therefore, to compare them to our proposed notions, namely IND-CCVA and SINT-PTXT, we make one small modification to existing definitions. Specifically, we allow each oracle used in the definitions to maintain states across invocations. It is easy to see that, this modification notwithstanding, the relations among existing notions shown in [4] and [5] remain the same. It is also easy to see that any schemes secure under the original definitions are secure under the definitions with this modification. Henceforth, we use the original names to refer to the modified definitions.

We provide the justifications of the relations in the full version of this paper [21]. We briefly discuss the relations shown in Figure 1 here. First, we comment that, as Figure 1 shows, SINT-PTXT is reasonably strong: it implies INT-PTXT but not the stronger notion of INT-CTXT. Also, an integrity notion, specifically INT-PTXT, turns out to be necessary for IND-CCVA, a privacy notion.

Being a necessary and sufficient characterization of secure encryption protocol of [9], IND-CCVA is not meant to constitute a complete security measure on its own. Rather, it guarantees secrecy only in conjunction with additional mechanisms that guarantee uniqueness of messages. Consequently, it may be surprising at first glance that IND-CCVA emerges as a notion that is incomparable to both IND-CPA and IND-CCA. In particular, IND-CCVA does not imply even a weak notion of privacy such as IND-CPA. Moreover, the proof of this relation can be easily extended to show that a channel protocol does not provide the stateful variant of semantic security either. (See the full version of this paper [21] for details.) The unfortunate implication here is that channel protocols proven secure as an encryption protocol may in fact leak information. This is a rather unexpected result since one would naturally assume that a secure encryption protocol should protect privacy of transmitted information. On the other hand, it is also arguably simply a technical issue that does not arise in many cases in practice. As pointed out in [9], if one can ensure that all messages are unique, then one can obtain security. One way to ensure uniqueness of messages is to simply prepend unique message IDs to all messages and to verify them when ciphertexts are received. In fact, many Internet protocols in use today (e.g., SSH, SSL, and TLS) already do so: they include in every packet a sequence number maintained internally by the communicating parties [15, 11, 23].

## Acknowledgments

I thank Mihir Bellare for his guidance and advice throughout the research and writing process for this paper. I also thank Ran Canetti and Hugo Krawczyk for their insights and comments especially regarding the notion of secure channels. Finally, I thank Tadayoshi Kohno, Bogdan Warinschi, and Alexandra Boldyreva for their helpful comments on earlier versions of this draft. The author is supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering and NSF CAREER Award CCR-9624439.

## References

1. R. Atkinson. Security architecture for the Internet protocol. RFC 1825, 1995.
2. M. Bellare, R. Canetti, and H. Krawczyk. Modular approach to the design and analysis of key exchange protocols. In *Proc. of the 30th ACM STOC*, pages 419–428, New York, NY, May 23–26 1998. ACM Press.
3. M. Bellare, A. Desai, E. Jorjipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proc. of the 38th FOCS*, pages 394–403. IEEE Computer Society Press, 1997.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *CRYPTO '98*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, August 1998.
5. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer-Verlag, December 2000.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. In Y. Desmedt, editor, *CRYPTO '94*, volume 839 of *LNCS*, pages 232–249. Springer-Verlag, 1994.
7. M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer-Verlag, December 2000.
8. M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In E. Brickell, editor, *CRYPTO '99*, volume 740 of *LNCS*, pages 519–536. Springer-Verlag, August 1999.
9. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 451–472. Springer-Verlag, 2001.
10. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer-Verlag, 2002.
11. T. Dierks and C. Allen. The TLS protocol: Version 1.0. RFC 2246, 1999.
12. W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, June 1992.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proc. of the 23rd ACM STOC*, pages 542–552, New Orleans, Louisiana, May 6–8 1991. ACM Press.

14. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.
15. A. Freier, P. Karlton, and P. Kocher. The SSL protocol: Version 3.0, 1996.
16. O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *J. Cryptology*, 6(1):21–53, 1993.
17. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.
18. C. Günther. An identity-based key-exchange protocol. In J.-J. Quisquater and J. Vandewille, editors, *EUROCRYPT '89*, volume 434 of *LNCS*, pages 29–37. Springer-Verlag, April 1990.
19. J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In B. Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 284–299. Springer-Verlag, April 2000.
20. H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 310–331. Springer-Verlag, August 2001.
21. C. Namprempre. Secure channels based on authenticated encryption schemes: A simple characterization. Cryptology ePrint Archive, Report 2002/065, May 2002. <http://eprint.iacr.org/>.
22. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO '91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, August 1991.
23. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH transport layer protocol, 2002. Draft, available at <http://www.ietf.org/internet-drafts/draft-ietf-secsh-transport-14.txt>.



# ID-Based Blind Signature and Ring Signature from Pairings

Fangguo Zhang and Kwangjo Kim

International Research center for Information Security (IRIS),  
Information and Communications University(ICU),  
58-4 Hwaam-dong Yusong-ku, Taejon, 305-732 Korea,  
{zhfg, kkj}@icu.ac.kr

**Abstract.** Recently the bilinear pairing such as Weil pairing or Tate pairing on elliptic curves and hyperelliptic curves have been found various applications in cryptography. Several identity-based (simply ID-based) cryptosystems using bilinear pairings of elliptic curves or hyperelliptic curves were presented. Blind signature and ring signature are very useful to provide the user's anonymity and the signer's privacy. They are playing an important role in building e-commerce. In this paper, we firstly propose an ID-based blind signature scheme and an ID-based ring signature scheme, both of which are based on the bilinear pairings. Also we analyze their security and efficiency.

**Keywords:** Blind signature, Ring signature, Bilinear pairings, ID-based cryptography, Provably security.

## 1 Introduction

False certification or no certification mechanisms cause problems, which can range from a “man-in-the-middle” attack (in order to gain knowledge over controlled data) to a completely open situation (to gain access to data and resources). It is important to note that these problems appear with encryption or even a secure protocol. If the user is led to connect to a spoofing site where appears to be what he wants, he may have a secure connection to a thief who will work maliciously. Thus, identity certification or authentication is necessary. In public key cryptosystem, each user has two keys, a private key and a public key. The binding between the public key (PK) and the identity (ID) of a user is obtained via a digital certificate. However, in a certificate-based system, before using the public key of a user, the participant must first verify the certificate of the user. As a consequence, this system requires a large amount of computing time and storage when the number of users increase rapidly. In 1984 Shamir [25] asked for ID-based encryption and signature schemes to simplify key management procedures in certificate-based public key setting. Since then, many ID-based encryption schemes and signature schemes [4][8][27] have been proposed.

The bilinear pairings, namely the Weil pairing and the Tate pairing of algebraic curves, are important tools for research on algebraic geometry. The early applications of the bilinear pairings in cryptography were used to evaluate the discrete logarithm problem. For example, the MOV attack [18] (using Weil pairing) and FR attack [9] (using Tate pairing) reduce the discrete logarithm problem on some elliptic curves or hyperelliptic curves to the discrete logarithm problem in a finite field. However, the bilinear pairings have been found various applications in cryptography recently [4] [5] [14] [15] [17] [23]. More precisely, they can be used to construct ID-based cryptographic schemes. Many ID-based cryptographic schemes have been proposed using the bilinear pairings. Examples are Boneh-Franklin's ID-based encryption scheme [4], Smart's ID-based authentication key agreement protocol [26], and several ID-based signatures schemes [6] [11] [20] [23], *etc.* The ID-based public key setting can be an alternative for certificate-based public key setting, especially when efficient key management and moderate security are required. In public key setting, users' anonymity is protected by means of blind signature, while signers' anonymity by group or ring signature. This paper is focused on ID-based blind signature and ID-based ring signature schemes.

The concept of blind signatures was introduced by Chaum [7], which provides anonymity of users in applications such as electronic voting and electronic payment systems, *etc.* In contrast to regular signature schemes, a blind signature scheme is an interactive two-party protocol between a user and a signer. It allows the user to obtain a signature of a message in a way that the signer learns neither the message nor the resulting signature. Blind signature plays a central role in building anonymous electronic cash.

Several ID-based signature schemes based on pairings were developed recently. In this paper, we propose a blind version of ID-based signature schemes. ID-based blind signature is attractive since one's public key is simply his identity. For example, if a bank issues electronic cash with ID-based blind signature, users and shops do not need to fetch bank's public key from a database. They can verify the electronic cash issued this year only by the following information, *Name of Country*  $\parallel$  *Name of City*  $\parallel$  *Name of Bank*  $\parallel$  *this year*.

The concept of ring signature was introduced by Rivest, Shamir and Tauman [22]. A ring signature is considered to be a simplified group signature which consists of only users without managers. It protects the anonymity of a signer since the verifier knows that the signature comes from a member of a ring, but doesn't know exactly who the signer is. There is also no way to revoke the anonymity of the signer. Ring signature can support *ad hoc* subset formation and in general does not require special setup. Rivest-Shamir-Tauman's ring signature scheme relies on general public-key setting.

After giving the formal definitions of ID-based blind signature and ring signature, we propose an ID-based blind signature scheme and an ID-based ring signature scheme using bilinear pairings, and analyze their security and efficiency.

**Organization of the Paper:** The rest of the paper is organized as follows: DLP, DDHP, CDHP, GDHP, and bilinear pairing are introduced in Section 2.

We give the formal definition of an ID-based blind signature scheme and an ID-based ring signature in Section 3. Our main ID-based blind signature scheme is presented in Section 4. Section 5 gives a security proof of our ID-based blind signature scheme. In Sections 6 and 7 we present an ID-based ring signature scheme and analyze its security and performance, respectively. Section 8 summarizes this paper and gives open problems.

## 2 Basic Concepts on Bilinear Pairings

Let  $G$  be a cyclic group generated by  $P$ , whose order is a prime  $q$ , and  $V$  be a cyclic multiplicative group of the same order  $q$ . The discrete logarithm problems in both  $G$  and  $V$  are hard. Let  $e : G \times G \rightarrow V$  be a pairing which satisfies the following conditions:

1. Bilinear:  $e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$  and  $e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)$ , or  $e(aP, bQ) = e(P, Q)^{ab}$ ;
2. Non-degenerate: There exists  $P \in G$  and  $Q \in G$  such that  $e(P, Q) \neq 1$ ;
3. Computability: There is an efficient algorithm to compute  $e(P, Q)$  for all  $P, Q \in G$ .

We note that the Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such bilinear maps.

Suppose that  $G$  is an additive group. Now we describe four mathematical problems.

- **Discrete Logarithm Problem (DLP)**: Given two group elements  $P$  and  $Q$ , find an integer  $n$ , such that  $Q = nP$  whenever such an integer exists.
- **Decision Diffie-Hellman Problem (DDHP)**: For  $a, b, c \in \mathbb{Z}_q^*$ , given  $P, aP, bP, cP$  decide whether  $c \equiv ab \pmod{q}$ .
- **Computational Diffie-Hellman Problem (CDHP)**: For  $a, b \in \mathbb{Z}_q^*$ , given  $P, aP, bP$ , compute  $abP$ .
- **Gap Diffie-Hellman Problem (GDHP)**: A class of problems where DDHP is easy while CDHP is hard.

We assume through this paper that CDHP and DLP are intractable, which means there is no polynomial time algorithm to solve CDHP or DLP with non-negligible probability. When the DDHP is easy but the CDHP is hard on the group  $G$ , we call  $G$  a *Gap Diffie-Hellman (GDH) group*. Such groups can be found on supersingular elliptic curves or hyperelliptic curves over finite field, and the bilinear parings can be derived from the Weil or Tate pairing  $e : G \times G \rightarrow V$ . Our schemes of this paper can be built on any GDH group.

## 3 Model

In this section, we give the formal definitions of ID-based blind signature scheme and ID-based ring signature scheme.

An ID-based blind signature scheme is considered be the combination of a general blind signature scheme and an ID-based one, *i.e.*, it is a blind signature, but its public key for verification is just the signer's identity.

**Definition 1 (ID-Based Blind Digital Signature).** *An ID-based blind signature scheme (simply IDBSS) consists of six-tuple (Trust Authority (or TA), Setup, User, Extract, Signer, Verification), where*

1. **TA** is a trustee which can issue a tamper-resistant equipment to transfer secret information to users. It executes two operations: System setup and User's private key generation.
2. **Setup** is a probabilistic polynomial algorithm that takes a security parameter  $k$ , and returns PARAMS (system parameters) and MASTER-KEY.
3. **Extract** is a probabilistic polynomial algorithm that takes as input PARAMS, MASTER-KEY and an arbitrary  $ID \in \{0, 1\}^*$ , and returns a private key  $S_{ID}$ . Here  $ID$  is a signer's identity and works as the signer's public key.
4. **Signer** and **User** are a pair of probabilistic interactive Turing machines, where both machines have the following tapes: a read-only input tape, a write-only output tape, a read/write working tape, a read-only random tape, and two communication tapes. **Signer** is given on its input tape  $(ID, S_{ID})$ . **User** is given on its input tape  $(ID, m)$ , where  $m$  is a message. The length of all input must be polynomial in  $k$ . **Signer** and **User** engage in the signature issuing protocol and stop in polynomial-time. At the end of this protocol, **Signer** outputs either completed or not-completed, and **User** outputs either fail or the signature  $\sigma(m)$  of the message  $m$ .
5. **Verification** is a probabilistic polynomial-time algorithm that takes  $(ID, m, \sigma(m))$  and outputs either accept or reject.

The security of an ID-based blind signature scheme consists of two requirements: the blindness property and the non-forgability of additional signatures. We say the blind signature scheme is secure if it satisfies two requirements.

Like [2] and [16], we give a formal definition of the blindness of ID-based blind signature scheme.

**Definition 2 (Blindness).** *Let  $\mathcal{A}$  be the Signer or a probabilistic polynomial-time algorithm that controls the Signer.  $\mathcal{A}$  is involved in the following game with two honest users, namely  $U_0$  and  $U_1$ .*

1.  $(ID, S_{ID}) \leftarrow \text{Extract}(\text{PARAMS}, ID)$ .
2.  $(m_0, m_1) \leftarrow \mathcal{A}(ID, S_{ID})$  ( $\mathcal{A}$  produces two documents).
3. Select  $b \in_R \{0, 1\}$  (*i.e.*,  $b$  is a random bit which is kept secret from  $\mathcal{A}$ ). Put  $m_b$  and  $m_{1-b}$  to the read-only input tape of  $U_0$  and  $U_1$ , respectively.
4.  $\mathcal{A}$  engages in the signature issuing protocol with  $U_0$  and  $U_1$  in arbitrary order.
5. If  $U_0$  and  $U_1$  output  $\sigma(m_b)$  and  $\sigma(m_{1-b})$ , respectively, on their private tapes, then give those outputs to  $\mathcal{A}$ . Otherwise, give  $\perp$  to  $\mathcal{A}$ .
6.  $\mathcal{A}$  outputs a bit  $b' \in \{0, 1\}$ .

If  $b' = b$ ,  $\mathcal{A}$  knows the message and its corresponding signature of each user. In this case, we say  $\mathcal{A}$  wins.

An ID-based signature is blind if, for all probabilistic polynomial-time algorithm  $\mathcal{A}$ ,  $\mathcal{A}$  wins in the following experiment with probability at most  $1/2 + 1/k^c$  for sufficiently large  $k$  and some constant  $c$ . The probability is taken over the coin flips of **Extract**, two users,  $U_0$  and  $U_1$ , and  $\mathcal{A}$ .

The ID-based ring signature can be viewed as the combination of a ring signature and an ID-based signature.

**Definition 3 (ID-Based Ring Digital Signature).** An ID-based ring signature scheme (simply IDRSS) consists of four-tuple, namely (**Setup**, **Extract**, **Signing**, **Verification**). Three parties are involved in the scheme: a **Signer**, a **User** and a **TA** (Like IDBSS, **TA** is a trustee, it executes two operations: System setup and User's private key generation).

1. **Setup** is a probabilistic polynomial algorithm, run by **TA**, that takes a security parameter  $k$  and returns PARAMS (system parameters) and MASTER-KEY.
2. **Extract** is a probabilistic polynomial algorithm, run by **TA**, that takes as input PARAMS, MASTER-KEY, and an arbitrary  $ID \in \{0,1\}^*$ . It returns a private key  $S_{ID}$ . Here  $ID$  is the signer's identity and used as the signer's public key.
3. **Signing** is a probabilistic polynomial algorithm that takes PARAMS, a private key  $S_{ID}$ , a list of identities,  $L$ , which includes  $ID$  corresponding to  $S_{ID}$ , and a message  $m$ . The algorithm outputs a signature  $\sigma(m)$  for  $m$ .
4. **Verification** is a probabilistic polynomial-time algorithm that takes  $(L, m, \sigma(m))$  and outputs either accept or reject.

We say an ID-based ring signature scheme is secure if it satisfies two requirements, namely, the unconditional ambiguity (i.e., the adversary cannot tell the identity of the signer with a probability larger than  $1/r$ , where  $r$  is the cardinality of the ring, even assuming that he/she has unlimited computing resources) and the non-forgeability of additional signatures.

## 4 Our ID-Based Blind Signature Scheme

In this section, we present an ID-based blind signature scheme from the bilinear pairings. Our scheme is similar to Schnorr's blind signature scheme.

Let  $G$  be a GDH group of prime order  $q$ . The bilinear pairing is given as  $e : G \times G \rightarrow V$ .

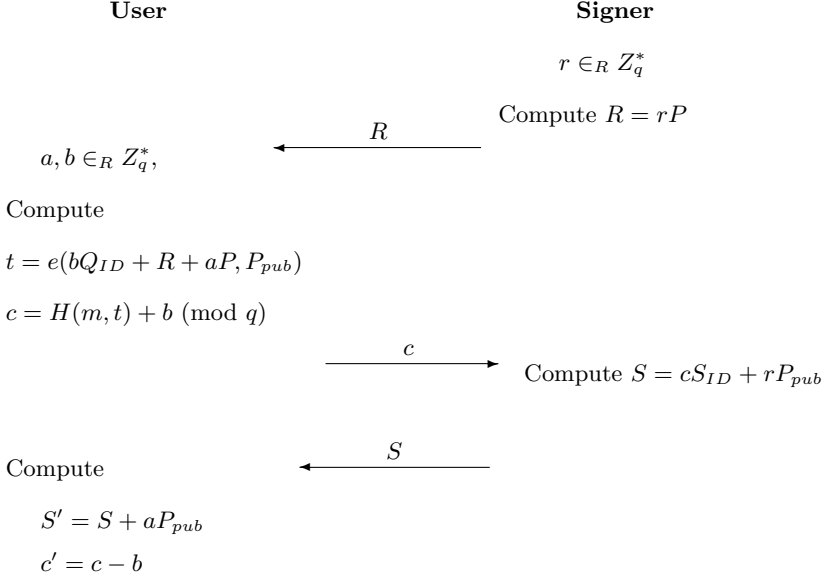
### [Setup]

Let  $P$  be a generator of  $G$ . Choose a random number  $s \in \mathbb{Z}_q^*$  and set  $P_{pub} = sP$ . Define two cryptographic hash functions  $H : \{0,1\}^* \rightarrow \mathbb{Z}/q$  and  $H_1 : \{0,1\}^* \rightarrow G$ . The system parameters are  $\text{PARAMS} = \{G, q, P, P_{pub}, H, H_1\}$ , and  $s$  be the MASTER-KEY of **TA**.

**[Extract]**

Given an identity  $ID$ , which implies the public key  $Q_{ID} = H_1(ID)$ , the algorithm returns the private key  $S_{ID} = sQ_{ID}$ .

The above two operations, **[Setup]** and **[Extract]** are carried out by TA. Note that TA can access to the sensitive private key  $S_{ID}$ . To avoid power abuse by TA,  $n$  trust authorities with  $(n, n)$ -threshold secret sharing scheme can be used to escrow the MASTER-KEY, as suggested in [11].

**[Blind Signature Issuing Protocol]**

**Fig. 1.** The blind signature issuing protocol

Suppose that  $m$  is the message to be signed. Let  $a \in_R$  denote the uniform random selection. The protocol is shown in Fig. 1.

- The signer randomly chooses a number  $r \in Z_q^*$ , computes  $R = rP$ , and sends  $R$  to the user as a commitment.
- (Blinding) The user randomly chooses  $a, b \in Z_q^*$  as blinding factors. He computes  $c = H(m, e(bQ_{ID} + R + aP, P_{pub})) + b \pmod{q}$ , and sends  $c$  to the signer.
- (Signing) The signer sends back  $S$ , where  $S = cS_{ID} + rP_{pub}$ .
- (Unblinding) The user computes  $S' = S + aP_{pub}$  and  $c' = c - b$ . He outputs  $\{m, S', c'\}$ .

Then  $(S', c')$  is the blind signature of the message  $m$ .

**[Verification:]**

Accept the signature if and only if

$$c' = H(m, e(S', P)e(Q_{ID}, P_{pub})^{-c'}).$$

To produce a blind signature, the **Signer** only requires to compute three scalar multiplications in  $G$ , while the **User** requires three scalar multiplications in  $G$ , one hash function evaluation and one bilinear pairing computation. The verification operation requires one hash function evaluation, two bilinear pairing computations and one exponentiation in  $V$ . One pairing computation can be saved, if a large number of verifications are to be performed for the same identity by precomputing  $e(Q_{ID}, P_{pub})$ . Our signature consists of an element in  $G$  and an element in  $V$ . In practice, the size of the element in  $G$  (elliptic curve group or hyperelliptic curve Jacobians) can be reduced by a factor of 2 with compression techniques in [12][13].

## 5 Analysis of the IDBSS

This section proves the security of our blind signature scheme assuming the intractability of CDHP and ideal randomness of hash functions  $H$  and  $H_1$ . For generic parallel attack, we assume the intractability of ROS-problem [24] i.e., to find an Overdetermined, Solveable system of linear equations modulo  $q$  with Random inhomogenities.

### 5.1 Correctness

The verification of the signature is justified by the following equations:

$$\begin{aligned}
 & H(m, e(S', P)e(Q_{ID}, P_{pub})^{-c'}) \\
 &= H(m, e(S + aP_{pub}, P)e(Q_{ID}, P_{pub})^{-c'}) \\
 &= H(m, e(cS_{ID} + rP_{pub} + aP_{pub}, P)e(Q_{ID}, P_{pub})^{-c'}) \\
 &= H(m, e(cS_{ID}, P)e(rP_{pub} + aP_{pub}, P)e(Q_{ID}, P_{pub})^{-c'}) \\
 &= H(m, e(S_{ID}, P)^c e((r + a)P_{pub}, P)e(Q_{ID}, P_{pub})^{-c'}) \\
 &= H(m, e(Q_{ID}, P_{pub})^c e((r + a)P, P_{pub})e(Q_{ID}, P_{pub})^{-c'}) \\
 &= H(m, e(Q_{ID}, P_{pub})^{c-c'} e(R + aP, P_{pub})) \\
 &= H(m, e(Q_{ID}, P_{pub})^b e(R + aP, P_{pub})) \\
 &= H(m, e(bQ_{ID} + R + aP, P_{pub})) \\
 &= H(m, t) = c - b = c'
 \end{aligned}$$

### 5.2 Security Proofs

On the blindness of our ID-based blind signature scheme, we can state the following theorem:

**Theorem 1.** *The proposed scheme is blind.*

*Proof.* We consider the experiment in Definition 2. Let  $\mathcal{A}$  be the **Signer** or a probabilistic polynomial-time algorithm that controls the **Signer** and has  $(ID, S_{ID})$  from **Extract**(PARAMS, ID).

If  $\mathcal{A}$  gets  $\perp$ , it is easy to see that  $\mathcal{A}$  wins the game with probability exactly the same as a random guessing of  $b$ , i.e., with probability  $1/2$ .

Suppose that  $\mathcal{A}$  gets  $\sigma(m_b)$  and  $\sigma(m_{1-b})$ , instead of  $\perp$ . For  $i = 0, 1$ , let  $R_i, c_i, S_i$  be the data exchanged during the signature issuing protocol, and  $(S'_0, c'_0)$  and  $(S'_1, c'_1)$  are given to  $\mathcal{A}$ . Then it is sufficient to show that there exist two random factors  $(\alpha, b)$  that map  $R_i, c_i, S_i$  to  $S'_j, c'_j$  for each  $i, j \in \{0, 1\}$  (here  $\alpha \in G$ ). We can define  $\alpha := S'_j - S_i, b := -c'_j - (-c_i)$ . As

$$e(R_i, P_{pub}) = e(S_i - c_i S_{ID}, P) = e(S_i, P) e(-c_i Q_{ID}, P_{pub}),$$

we have:

$$\begin{aligned} c'_j &= H(m, e(S'_j, P) e(Q_{ID}, P_{pub})^{-c'_j}) \\ &= H(m, e(S_i + \alpha, P) e(Q_{ID}, P_{pub})^{b-c_i}) \\ &= H(m, e(c_i Q_{ID} + R_i, P_{pub}) e(\alpha, P) e(Q_{ID}, P_{pub})^{b-c_i}) \\ &= H(m, e(R_i, P_{pub}) e(\alpha, P) e(Q_{ID}, P_{pub})^b) \end{aligned}$$

Thus the blinding factors always exist which lead to the same relation defined in the signature issuing protocol. Therefore, even an infinitely powerful  $\mathcal{A}$  succeeds in determining  $b$  with probability  $\frac{1}{2}$ .

Taking two cases into account, the probability that  $\mathcal{A}$  wins is  $\frac{1}{2}$ . Therefore, the proposed scheme is blind.  $\square$

Next, we discuss the non-forgeability of the proposed ID-based blind signature scheme. Let  $\mathcal{A}$  be the adversary who controls **User**. We consider three cases.

### Case 1: Non-interaction with Signer

If  $\mathcal{A}$  successful produces a valid message-signature pairing  $(m, \sigma(m))$  with a non-negligible probability  $\eta$ , then we will show that using  $\mathcal{A}$ , we can construct a simulator  $\mathcal{M}$  to solve the CDHP with the non-negligible probability  $\eta$ .

Let  $q_H$  be the maximum number of queries asked from  $\mathcal{A}$  to  $H$ , it is limited by a polynomial in  $k$ . We assume that all queries are different. Let  $(G, V, q, e(\cdot, \cdot), P, P_{pub}, Q_{ID})$  be the problem that we want to solve: to find  $S_{ID} \in G$  from  $e(Q_{ID}, P_{pub}) = e(S_{ID}, P)$ .  $\mathcal{M}$  simulates as follows:

- Select  $I \in_R \{1, \dots, q_H\}$ .
- Let  $\mathcal{A}$  simulates  $H$  as follows: For  $i$ -th query to  $H$ , if  $i = I$ , then ask  $H$  for the answer. Otherwise, randomly select and output an element from  $Z_q$ .
- Randomly input a number  $r \in Z_q$ , send  $R = rP$  to  $\mathcal{A}$ .
- $\mathcal{A}$  outputs a signature  $(m_I, S', c')$ .



We denote by  $\eta$  the success probability of  $\mathcal{M}$ , which is non-negligible.

Now we use  $\mathcal{M}$  to get  $S_{ID}$  from  $e(Q_{ID}, P_{pub}) = e(S_{ID}, P)$ . We run  $\mathcal{M}$  with a random tape (*i.e.*, with random input  $(a, b, r)$  and a random choice of  $H$ ).  $\mathcal{M}$  then outputs a valid signature  $(S'_1, c'_1)$  after trying  $1/\eta$  times. We rewind  $\mathcal{M}$  with the same random tape and run it with a different choice of  $H$ . After at most  $2/\eta$  times, we can get another valid signature  $(S'_2, c'_2)$ . Then we have

$$S'_1 - S'_2 = c'_1 S_{ID} - c'_2 S_{ID},$$

Because  $c'_1$  and  $c'_2$  are different choices of  $H$ , *i.e.*,  $c'_1 \neq c'_2$ , we can get  $S_{ID} = ((c'_1 - c'_2)^{-1} \bmod q)(S'_1 - S'_2)$ . If  $P_{pub} = sP$ ,  $Q_{ID} = H_1(ID) = tP$ , then  $S_{ID} = stP$ , *i.e.*, we solved CDHP.

Since we assume that the CDHP is intractable, the success probability of the forgery in this case is negligible.

### Case 2: Non-fixed ID Forgery

We assume that **Extract** is a random oracle, and allow an adversary  $\mathcal{A}$  to query it.  $\mathcal{A}$  executes the following experiment:

1.  $(ID, S_{ID}) \leftarrow \mathbf{Extract}(\text{PARAMS}, ID)$ .
2.  $\mathcal{A}$  queries **Extract**  $q_E$  ( $q_E > 0$ ) times with  $(\text{PARAMS}, ID_i \neq ID)$  for  $i = 1, \dots, q_E$ . **Extract** returns to  $\mathcal{A}$  the  $q_E$  corresponding secret key  $S_{ID_i}$ . We assume that  $q_E$  is limited by a polynomial in  $k$ .
3.  $\mathcal{A}$  produces  $q_E$  signatures with the help of  $(ID_i, S_{ID_i})$ .
4.  $\mathcal{A}$  outputs a signature  $(m, \sigma(m))$ .

Since  $H$  and  $H_1$  are random oracles, both **Extract** and the blind signature issuing protocol between **User** and **Signer** generate random numbers with uniform distributions. This means that  $\mathcal{A}$  learns nothing from query results. Case 2 can be reduced to Case 1, so we claim that, under the argument that all hash functions are random oracles and that the CDHP is intractable, the successful probability of the non-fixed ID attack on the proposed scheme is negligible.

### Case 3: Fixed ID Generic Parallel Attack

In [24], Schnorr proposed a new attack, called *generic parallel attack*, on Schnorr's blind signature scheme. This attack also applies to our blind scheme. In the following, we prove that our scheme is secure against the generic parallel attack under the assumption of the intractability of the ROS-problem.

We first describe how  $\mathcal{A}$  uses the generic parallel attack to forge  $l + 1$  valid ID-based blind signatures in our scheme. Let  $q_H$  be the maximum number of queries of  $H$  from  $\mathcal{A}$ .

1. The signer sends commitments  $R_1 = r_1P, R_2 = r_2P, \dots, R_l = r_lP$ .
2.  $\mathcal{A}$  selects randomly  $a_{k,1}, a_{k,2}, \dots, a_{k,l} \in Z_q$  and messages  $m_1, m_2, \dots, m_t$ . He computes  $f_k = e(\sum_{i=1}^l a_{k,i} R_i, P_{pub})$  and  $H(m_k, f_k)$  for  $k = 1, 2, \dots, t$ . Here  $t < q_H$ .

3.  $\mathcal{A}$  solves  $l + 1$  of  $t$  Eqs. (II) in the unknowns  $c_1, c_2, \dots, c_l$  over  $Z_q$  :

$$H(m_k, f_k) = \sum_{j=1}^l a_{k,j} c_j \quad \text{for } k = 1, 2, \dots, t. \quad (1)$$

4.  $\mathcal{A}$  sends the solutions  $c_1, c_2, \dots, c_l$  as challenge to the signer.

5. The signer sends back  $S_i = c_i S_{ID} + r_i P_{pub}$  for  $i = 1, 2, \dots, l$ .

6. For each solved Eq. (II),  $\mathcal{A}$  gets a valid signature  $(m_k, S'_k, c'_k)$  by setting

$$c'_k := \sum_{j=1}^l a_{k,j} c_j = H(m_k, f_k)$$

and

$$S'_k := \sum_{j=1}^l a_{k,j} S_j.$$

7.  $\mathcal{A}$  outputs  $l + 1$  signatures  $(m_k, S'_k, c'_k)$  for  $k = 1, 2, \dots, l + 1$ .

It is easy to see that the forged signature is valid. According to Eq. (II), we have:

$$\begin{aligned} e(S'_k, P) e(Q_{ID}, P_{pub})^{-c'_k} &= e \left( \sum_{j=1}^l a_{k,j} S_j, P \right) e(Q_{ID}, P_{pub})^{-c'_k} \\ &= e \left( \sum_{j=1}^l a_{k,j} (c_j S_{ID} + r_j P_{pub}), P \right) e(Q_{ID}, P_{pub})^{-c'_k} \\ &= e(S_{ID}, P)^{\sum_{j=1}^l a_{k,j} c_j} e \left( \sum_{j=1}^l a_{k,j} r_j P_{pub}, P \right) e(Q_{ID}, P_{pub})^{-c'_k} \\ &= e \left( \sum_{j=1}^l a_{k,j} R_j, P_{pub} \right) = f_k \end{aligned}$$

and

$$H(m_k, e(S'_k, P) e(Q_{ID}, P_{pub})^{-c'_k}) = c'_k$$

The essence of the above attack is to solve the so-called ROS-problem, which is shown below.

**ROS-Problem**[24]: Find an overdetermined, solveable system of linear equations modulo  $q$  with random inhomogenities. More precisely, given an oracle random function  $F : Z_q^l \rightarrow Z_q$ , find coefficients  $a_{k,i} \in Z_q$  and a solvable system of  $l + 1$  distinct equations of Eq. (2) in the unknowns  $c_1, c_2, \dots, c_l$  over  $Z_q$  :

$$a_{k,1} c_1 + \dots + a_{k,l} c_l = F(a_{k,1}, \dots, a_{k,l}) \quad \text{for } k = 1, 2, \dots, t. \quad (2)$$

The security against the *generic parallel attack* to our ID-based blind signature scheme depends on the difficulty of ROS-problem. As Schnorr states that the intractability of the ROS-problem is “a palausible but novel complexity assumption”. At Crypto2002, D. Wagner [28] claimed that he can break ROS-problem with subexponential time. To be resistant against this new attack,  $q$  may need to be at least 1600 bits long.

**Remark:** The most powerful attack on blind signature is the *one-more signature forgery* introduced by Pointcheval and Stern in [21]. They suggested two kinds of attacks: the sequential attack and the parallel attack. But at the moment we believe that their method can't be applied to our scheme, since multiple key components involve their blind signature scheme, while only one single private key is engaged in our scheme. Schnorr [24] proved that the security against the *one-more signature forgery* of his blind signature scheme depends on the difficulty of ROS-problem. However, our ID-based blind signature scheme seems difficult to prove that the security against the sequential *one-more signature forgery* depends on the difficulty of ROS-problem. We remain an open problem to find a formal proof against the sequential *one-more signature forgery* on our scheme.

## 6 Our ID-Based Ring Signature Scheme

The concept of ring signature has recently been formalized by Rivest *et al.* in [22]. A ring signature allows a member of an *ad hoc* collection of users  $U$  to prove that a message is authenticated by a member  $U$ . It is very useful in anonymity protection. Naor [19] combined the deniable authentication and Rivest *et al.*'s ring signature and proposed *Deniable Ring Authentication*.

The first ring signature scheme is based on RSA cryptosystem and the general certificate-based public key setting. The first ring signature scheme based on DLP was proposed by M. Abe, M. Ohkubo, and K. Suzuki in [1] recently, and their scheme is based on the general certificate-based public key setting too. In this section, we present an ID-based ring signature scheme using pairings.

Let  $G$  be a GDH group of prime order  $q$ . The bilinear pairing is  $e : G \times G \rightarrow V$ .

### [Setup]

The system setup is the same as IDBSS. The system parameters  $\text{PARAMS} = \{G, q, P, P_{\text{pub}}, H, H_1\}$ . The master key of TA is  $s$ .

### [Extract]

Given an identity  $ID$ , the algorithm outputs  $S_{ID} = sH_1(ID)$  as the private key associated with  $ID$ . The public key is given by  $Q_{ID} = H_1(ID)$ .

Let  $ID_i$  be a user's identity, and  $S_{ID_i}$  be the private key associated with  $ID_i$  for  $i = 1, 2, \dots, n$ . Let  $L = \{ID_i\}$  be the set of identities. The real signer's identity  $ID_k$  is listed in  $L$ .

**[Signing]**

- (Initialization): Choose randomly an element  $A \in G$ , compute  $c_{k+1} = H(L \parallel m \parallel e(A, P))$ .
- (Generate forward ring sequence): For  $i = k + 1, \dots, n - 1, 0, 1, \dots, k - 1$  (i.e., the value of  $i$  all modulo  $n$ ), choose randomly  $T_i \in G$  and compute  $c_{i+1} = H(L \parallel m \parallel e(T_i, P)e(c_i H_1(ID_i), P_{pub}))$ .
- (Forming the ring): Compute  $T_k = A - c_k S_{ID_k}$ .
- (Output the ring signature): Select 0 (i.e.,  $n$ ) as the glue value, the resulting signature for  $m$  and  $L$  is the  $(n + 1)$ -tuple:  $(c_0, T_0, T_1, \dots, T_{n-1})$ .

**[Verification]**

Given  $(c_0, T_0, T_1, \dots, T_{n-1})$ ,  $m$ , and  $L$ , compute

$$c_{i+1} = H(L \parallel m \parallel e(T_i, P)e(c_i H_1(ID_i), P_{pub})) \text{ for } i = 0, 1, \dots, n - 1.$$

Accept if  $c_n = c_0$ , and reject otherwise.

## 7 Analysis of the IDRSS

### 7.1 Correctness

From the procedure of ring signature generation, we have:

$$\begin{aligned}
 c_{k+1} &= H(L \parallel m \parallel e(A, P)) \\
 c_{k+2} &= H(L \parallel m \parallel e(T_{k+1}, P)e(c_{k+1} H_1(ID_{k+1}), P_{pub})) \\
 &\vdots \\
 c_n &= H(L \parallel m \parallel e(T_{n-1}, P)e(c_{n-1} H_1(ID_{n-1}), P_{pub})) \\
 &= c_0 \\
 c_1 &= H(L \parallel m \parallel e(T_0, P)e(c_0 H_1(ID_0), P_{pub})) \\
 c_2 &= H(L \parallel m \parallel e(T_1, P)e(c_1 H_1(ID_1), P_{pub})) \\
 &\vdots \\
 c_k &= H(L \parallel m \parallel e(T_{k-1}, P)e(c_{k-1} H_1(ID_{k-1}), P_{pub}))
 \end{aligned}$$

Since  $T_k = A - c_k S_{ID_k}$ , in the procedure of ring signature verification, we have:

$$\begin{aligned}
 c_{k+1} &= H(L \parallel m \parallel e(T_k, P)e(c_k H_1(ID_i), P_{pub})) \\
 &= H(L \parallel m \parallel e(A - c_k S_{ID_k}, P)e(c_k H_1(ID_i), P_{pub})) \\
 &= H(L \parallel m \parallel e(A, P)e(-c_k S_{ID_k}, P)e(c_k H_1(ID_i), P_{pub})) \\
 &= H(L \parallel m \parallel e(A, P)e(-c_k H_1(ID_i) + c_k H_1(ID_i), P_{pub})) \\
 &= H(L \parallel m \parallel e(A, P))
 \end{aligned}$$

The sequence  $\{c_i\}$  ( $i = 0, 1, \dots, n - 1$ ) in the ring signature verification procedure is the same as the ring signature generation procedure, so we have  $c_n = c_0$ .

## 7.2 Security

Our ID-based ring scheme holds unconditionally signer-ambiguity, because all  $T_i$  but  $T_k$  are taken randomly from  $G$ . In fact, at the starting point, the  $T_k$  is also distributed uniformly over  $G$ , since  $A$  is randomly chosen from  $G$ . Therefore, for fixed  $L$  and  $m$ ,  $(T_0, T_1, \dots, T_{n-1})$  has  $|G|^n$  solutions, all of which can be chosen by the signature generation procedure with equal probability, regardless of the signer.

When  $n = 1$ , our ID-based ring signature reduces to the ID-based signature scheme proposed by F. Hess [11] (Let  $P_1 = P_{pub}$  in Hess scheme). Hess's ID-based signature scheme is non-forgeability under the assumption of the intractability of the CDHP and all hash functions are random oracles.

For  $n > 1$ , we fix a set of identities, denoted by  $L$ . Suppose that  $\mathcal{A}$ 's identity  $ID_A$  is not listed in  $L$ , but he wants to forge a valid ring signature.  $\mathcal{A}$  can either forge a valid signature of a user whose identity  $ID_k$  is listed in  $L$  (this is the same as the case of  $n = 1$ ), or executes the following experiment:

- S1  $\mathcal{A}$  queries **Extract**  $q_E$  ( $q_E > 0$ ) times with  $(PARAMS, ID_i \notin L)$  for  $i = 1, \dots, q_E$ . **Extract** returns to  $\mathcal{A}$  the  $q_E$  corresponding secret key  $S_{ID_i}$ .
- S2 Choose randomly an integer  $c_0 \in \mathbb{Z}_q$ .
- S3 Do the same as "generate forward ring sequence" of [**Signing**] for  $i = 0, 1, \dots, n - 2$ , where  $n = |L|$ .
- S4 Assign  $c_0$  to  $H(L \parallel m \parallel e(T_{n-1}, P)e(c_{n-1}H_1(ID_{n-1}), P_{pub}))$ .
- S5 Output the ring signature:  $(c_0, T_0, T_1, \dots, T_{n-1})$ .

If  $\mathcal{A}$  finishes above S1 and get a  $(ID'_i, S_{ID'_i})$ , such that  $H_1(ID'_i) = H_1(ID_j)$ ,  $ID_j \in L$ , then he can forge a valid ring signature. But since  $H_1$  is random oracle, **Extract** generates random numbers with uniform distributions. This means that  $\mathcal{A}$  learns nothing from query results. Since  $H$  is acted as a random oracle too and all  $T_i$  are taken randomly from  $G$ , the probability of  $c_0 = H(L \parallel m \parallel e(T_{n-1}, P)e(c_{n-1}H_1(ID_{n-1}), P_{pub}))$  is  $1/q$ . So we say that the proposed ID-based ring signature scheme is non-forgeable.

## 7.3 Efficiency

Our ring signature scheme can be performed with supersingular elliptic curves or hyperelliptic curves. The essential operation in our ID-based signature schemes is to compute a bilinear pairing. Due to [3] and [10], the computation of a bilinear pairing becomes efficient. Furthermore, the length of signature can be reduced by a factor of 2 using compression technique.

Since our scheme is based on identity rather than an arbitrary number, a public key consists of some aspects of a user's information which may uniquely identify himself, such as email address. In some applications, the lengths of public keys and signatures can be reduced. For instance, in an electronic voting or an electronic auction system, the registration manager (RM) can play the role of TA in an ID-based cryptosystem. In the registration phase, RM gives a bidder or a voter his registration number as his public key =  $\{(The\ name\ of\ the\ e-voting$

or *e-auction system* || *RM* || *Date* || *Number*),  $n$  }. Here  $n$  is the number of all bidders or voters.

## 8 Summary and Open Problems

The ID-based public key setting can be an alternative for certificate-based public key setting, when efficient key management and moderate security are required in particular. In this paper, we proposed an ID-based blind signature scheme and ID-based ring signature scheme using the bilinear pairing. We also analyzed their security and efficiency. Our ID-based blind signature scheme and ID-based ring signature scheme can be easily combined to design electronic voting scheme or electronic cash scheme.

The security of our ID-based blind signature scheme against the *generic parallel attack* depends on the difficulty of ROS-problem. At Crypto2002, D. Wagner [28] claimed that he can break ROS-problem with subexponential time. To be resistant against this new attack,  $q$  may need to be at least 1600 bits long. Our ID-based blind signature scheme maybe not so efficient in implementation. To improve our ID-based blind signature scheme against the *generic parallel attack* remains as an open problem. On the security against the sequential *one-more signature forgery* of our ID-based blind signature scheme, we expect to find a formal proof under standard assumptions.

## Acknowledgements

The authors are grateful to the anonymous reviewers for their valuable suggestions and comments on this paper.

## References

1. M. Abe, M. Ohkubo, and K. Suzuki, *1-out-of- $n$  signatures from a variety of keys*, To appear in Advances in Cryptology-Asiacrypt 2002, 2002.
2. M. Abe and T. Okamoto, *Provably secure partially blind signatures*, Advances in Cryptology-Crypto 2000, LNCS 1880, pp.271-286, Springer-Verlag, 2000.
3. P.S.L.M. Barreto, H.Y. Kim, B.Lynn, and M.Scott, *Efficient algorithms for pairing-based cryptosystems*, Advances in Cryptology-Crypto 2002, LNCS 2442, pp.354-368, Springer-Verlag, 2002.
4. D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Advances in Cryptology-Crypto 2001, LNCS 2139, pp.213-229, Springer-Verlag, 2001.
5. D. Boneh, B. Lynn, and H. Shacham, *Short signatures from the Weil pairing*, Advances in Cryptology-Asiacrypt 2001, LNCS 2248, pp.514-532, Springer-Verlag, 2001.
6. J.C. Cha and J.H. Cheon, *An identity-based signature from gap Diffie-Hellman groups*, Cryptology ePrint Archive, Report 2002/018, available at <http://eprint.iacr.org/2002/018/>.
7. D. Chaum, *Blind signatures for untraceable payments*, Advances in Cryptology-Crypto 82, Plenum, NY, pp.199-203, 1983.

8. C.Cocks, *An identity based encryption scheme based on quadratic residues*, In Cryptography and Coding, LNCS 2260, pp.360-363, Springer-Verlag, 2001.
9. G. Frey and H.Rück, *A remark concerning  $m$ -divisibility and the discrete logarithm in the divisor class group of curves*, Mathematics of Computation, 62, pp.865-874, 1994.
10. S. D. Galbraith, K. Harrison, and D. Soldera, *Implementing the Tate pairing*, ANTS 2002, LNCS 2369, pp.324-337, Springer-Verlag, 2002.
11. F. Hess, *Exponent group signature schemes and efficient identity based signatureschemes based on pairings*, Cryptology ePrint Archive, Report 2002/012, available at <http://eprint.iacr.org/2002/012/>.
12. F. Hess G. Seroussi and N. Smart, *Two topics in hyperelliptic cryptography*, SAC (Selected Areas in Cryptography) 2001, LNCS 2259, pp.181-189, Springer-Verlag, 2001.
13. IEEE Std 2000-1363, *Standard specifications for public key cryptography*, 2000.
14. A. Joux, *A one round protocol for tripartite Diffie-Hellman*, ANTS IV, LNCS 1838, pp.385-394, Springer-Verlag, 2000.
15. A. Joux, *The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems*, ANTS 2002, LNCS 2369, pp.20-32, Springer-Verlag, 2002.
16. A. Juels, M. Luby and R. Ostrovsky, *Security of blind digital signatures*, Advances in Cryptology-Crypto 97, LNCS 1294, pp.150-164, Springer-Verlag, 1997.
17. M.S. Kim and K. Kim, *A new identification scheme based on the bilinear Diffie-Hellman problem*, Proc. of ACISP(The 7th Australasian Conference on Information Security and Privacy) 2002, LNCS 2384, pp.464-481, Springer-Verlag, 2002.
18. A. Menezes, T. Okamoto, and S. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transaction on Information Theory, Vol.39, pp.1639-1646, 1993.
19. M. Naor, *Deniable Ring Authentication*, Advances in Cryptology-Crypto 2002, LNCS 2442, pp.481-498, Springer-Verlag, 2002.
20. K.G. Paterson, *ID-based signatures from pairings on elliptic curves*, Cryptology ePrint Archive, Report 2002/004, available at <http://eprint.iacr.org/2002/004/>.
21. D. Pointcheval and J. Stern, *Security arguments for digital signatures and blind signatures*, Journal of Cryptology, Vol.13, No.3, pp.361-396, 2000.
22. R.L. Rivest, A. Shamir and Y. Tauman, *How to leak a secret*, Advances in Cryptology-Asiacrypt 2001, LNCS 2248, pp.552-565, Springer-Verlag, 2001.
23. R. Sakai, K. Ohgishi, M. Kasahara, *Cryptosystems based on pairing*, SCIS 2000-C20, Okinawa, Japan. Jan. 2000.
24. C. P. Schnorr, *Security of blind discrete log signatures against interactive attacks*, ICICS 2001, LNCS 2229, pp. 1-12, Springer-Verlag, 2001.
25. A. Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology-Crypto 84, LNCS 196, pp.47-53, Springer-Verlag, 1984.
26. N.P. Smart, *Identity-based authenticated key agreement protocol based on Weil pairing*, Electron. Lett., Vol.38, No.13, pp.630-632, 2002.
27. S. Tsuji and T.Itoh, *An ID-based cryptosystem based on the discrete logarithm problem*, IEEE Journal of Selected Areas in Communications, Vol.7, No.4, pp.467-473, 1989.
28. D. Wagner, *A generalized birthday problem*, Advances in Cryptology-Crypto 2002, LNCS 2442, pp.288-303, Springer-Verlag, 2002.

# Hierarchical ID-Based Cryptography

Craig Gentry<sup>1</sup> and Alice Silverberg<sup>2,\*</sup>

<sup>1</sup> DoCoMo USA Labs, San Jose, CA, USA,

`cgentry@docomolabs-usa.com`

<sup>2</sup> Department of Mathematics, Ohio State University, Columbus, OH, USA,

`silver@math.ohio-state.edu`

**Abstract.** We present hierarchical identity-based encryption schemes and signature schemes that have total collusion resistance on an arbitrary number of levels and that have chosen ciphertext security in the random oracle model assuming the difficulty of the Bilinear Diffie-Hellman problem.

**Keywords:** identity-based cryptography, hierarchical identity-based cryptography, elliptic curves, pairing-based cryptography

## 1 Introduction

Our main result is an efficient construction for accomplishing hierarchical identity-based encryption while retaining total collusion resistance (in the random oracle model). Prior to this paper, the only method known for making identity-based encryption hierarchical substantially sacrificed security (collusion resistance) or efficiency.

### 1.1 Identity-Based Encryption

In traditional public key encryption, Bob's public key is a random string unrelated to his identity. When Alice wants to send a message to Bob, she must first obtain Bob's authenticated public key. Typical solutions to this problem involve public key directories. The main idea in identity-based encryption is to eliminate the public key distribution problem by making Bob's public key derivable from some known aspect of his identity, such as his email address. When Alice wants to send a message to Bob, she merely derives Bob's public key directly from his identifying information. Public key directories are unnecessary.

Shamir [17] proposed the idea of identity-based cryptography in 1984, and described an identity-based signature scheme in the same article. However, practical identity-based encryption (IBE) schemes were not found until recently with the work of Boneh and Franklin [5,6] and Cocks [8] in 2001. Cocks's scheme is

---

\* Silverberg would like to thank DoCoMo USA Labs for support and kind hospitality during her visit there.



based on the Quadratic Residuosity Problem, and although encryption and decryption are reasonably fast (about the speed of RSA), there is significant message expansion, i.e., the bit-length of the ciphertext is many times the bit-length of the plaintext. The Boneh-Franklin scheme bases its security on the Bilinear Diffie-Hellman Problem, and is quite fast and efficient when using Weil or Tate pairings on supersingular elliptic curves or abelian varieties.

We must note that ID-based encryption has some disadvantages. Bob receives his private key from a third party called a Private Key Generator (PKG) that computes his private key as a function of its master secret and Bob's identity. This requires Bob to authenticate himself to the PKG (in the same way he would authenticate himself to a CA), and requires a secure channel through which the PKG may send Bob his private key. Bob's PKG must publish parameters that embed its master secret, and Alice must obtain these parameters before sending an encrypted message to Bob. Another disadvantage is that the PKG knows Bob's private key, i.e., key escrow is inherent in ID-based systems. Clearly, escrow is a serious problem for some applications.

However, the advantages of identity-based encryption are compelling. The problem of obtaining authentic public keys has been replaced by the problem of obtaining authentic public parameters of PKGs, but the latter should be less burdensome since there will be substantially fewer PKGs than total users. For example, if everyone uses a single PKG, then everyone in the system can communicate securely without ever having to perform online lookup of public keys or public parameters.

## 1.2 Motivation for Hierarchical ID-Based Encryption (HIDE)

Although having a single PKG would completely eliminate online lookup, it is undesirable for a large network because the PKG becomes a bottleneck. Not only is private key generation computationally expensive, but also the PKG must verify proofs of identity and must establish secure channels to transmit private keys. Hierarchical ID-based encryption (HIDE) allows a root PKG to distribute the workload by delegating private key generation and identity authentication to lower-level PKGs. In a HIDE scheme, a root PKG need only generate private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level. Authentication and private key transmission can be done locally. To encrypt a message to Bob, Alice need only obtain the public parameters of Bob's *root* PKG (and Bob's identifying information); there are no "lower-level parameters." Another advantage of HIDE schemes is damage control: disclosure of a domain PKG's secret does not compromise the secrets of higher-level PKGs. The schemes of Cocks and Boneh-Franklin do not have these properties.

A hierarchical ID-based key sharing scheme with partial collusion-resistance is given in [10,11]. Horwitz and Lynn [12] introduced hierarchical identity-based encryption, and proposed a 2-level HIDE scheme with total collusion-resistance at the first level and with partial collusion-resistance at the second level, i.e., (a threshold number of) users can collude to obtain the secret of their domain PKG

(and thereafter masquerade as the domain PKG). This scheme may be practical for applications where collusion below the first level is not a concern. Finding a secure and practical hierarchical identity-based encryption scheme was, prior to this paper, an important open question.

### 1.3 Our Results

The scheme in this paper extends the Boneh-Franklin IBE scheme in a natural way. It is a practical, fully scalable, HIDE scheme with total collusion resistance and chosen ciphertext security in the random oracle model, regardless of the number of levels in the hierarchy, assuming the difficulty of the same Bilinear Diffie-Hellman (BDH) problem given in [6] (see Section 2 below). The scheme is quite efficient – the bit-length of the ciphertext and the complexity of decryption grow only linearly with the level of the message recipient.<sup>1</sup> For example, if Bob is at level 1 (just below the root PKG) and Carol is at level 10, Alice’s ciphertext to Carol will be about 10 times as long as Alice’s ciphertext to Bob, and Carol will take about 10 times as long as Bob to decrypt the message from Alice. At the top level, our HIDE scheme is as fast and efficient as Boneh-Franklin. We show how the scheme can be modified to reduce ciphertext expansion.

The intuitively surprising aspect of this scheme is that, even though lower-level PKGs generate additional random information, this does not necessitate adding public parameters below the root level. Also, the random information generated by a lower-level PKG does not adversely affect the ability of users not under the lower-level PKG to send encrypted communications to users under the lower-level PKG.

A hierarchical ID-based signature (HIDS) scheme follows naturally from our HIDE scheme (see Section 4). We also introduce the concept of dual-ID-based encryption (where the ciphertext is a function of both the encrypter and decrypter’s identities) and show how this concept, in the context of hierarchical ID-based encryption, allows the length of the ciphertext to be reduced and permits the creation of “escrow shelters” that limit the scope of key escrow.

The rest of the paper is organized as follows. Definitions and background information are given in Section 2. Our hierarchical ID-based encryption scheme is presented in Section 3. An associated hierarchical ID-based signature scheme is given in Section 4. Section 5 gives modifications to minimize the ciphertext expansion. Section 6 discusses how to restrict the scope of key escrow. Section 7 gives security definitions and results (the full version will contain the proofs). Additional extensions and variations are given in Section 8.

## 2 Definitions

In this section, we give some definitions similar to those given in [5, 6, 12].

---

<sup>1</sup> Contrast this with [12], where the complexity of encryption grows linearly with the *security* against collusion of a domain PKG’s secret. Our scheme has total collusion resistance assuming the difficulty of BDH.

**ID-Tuple:** A user has a position in the hierarchy, defined by its tuple of IDs:  $(ID_1, \dots, ID_t)$ . The user's ancestors in the hierarchy tree are the root PKG and the users/lower-level PKGs whose ID-tuples are  $\{(ID_1, \dots, ID_i) : 1 \leq i < t\}$ .

**Hierarchical Identity-Based Encryption (HIDE):** a HIDE scheme is specified by five randomized algorithms: Root Setup, Lower-level Setup, Extraction, Encryption, and Decryption:

**Root Setup:** The root PKG takes a security parameter  $K$  and returns *params* (system parameters) and a root secret. The system parameters include a description of the message space  $\mathcal{M}$  and the ciphertext space  $\mathcal{C}$ . The system parameters will be publicly available, while only the root PKG will know the root secret.

**Lower-Level Setup:** Lower-level users must obtain the system parameters of the root PKG. In HIDE schemes, a lower-level user is not permitted to have any “lower-level parameters” of its own. However, this constraint does not necessarily preclude a lower-level PKG from generating its own lower-level secret, which it may use in issuing private keys to its children. In fact, in our HIDE scheme, a lower-level PKG may generate a lower-level secret, or it may generate random one-time secrets for each Extraction.

**Extraction:** A PKG (whether the root one or a lower-level one) with ID-tuple  $(ID_1, \dots, ID_t)$  may compute a private key for any of its children (e.g., with ID-tuple  $(ID_1, \dots, ID_t, ID_{t+1})$ ) by using the system parameters and its private key (and any other secret information).

**Encryption:** A sender inputs *params*,  $M \in \mathcal{M}$  and the ID-tuple of the intended message recipient, and computes a ciphertext  $C \in \mathcal{C}$ .

**Decryption:** A user inputs *params*,  $C \in \mathcal{C}$ , and its private key  $d$ , and returns the message  $M \in \mathcal{M}$ .

Encryption and decryption must satisfy the standard consistency constraint, namely when  $d$  is the private key generated by the Extraction algorithm for ID-tuple, then:

$$\forall M \in \mathcal{M} : \text{Decryption}(\text{params}, d, C) = M$$

where  $C = \text{Encryption}(\text{params}, \text{ID-tuple}, M)$ .

**Hierarchical ID-Based Signature (HIDS):** a HIDS scheme is specified by five randomized algorithms: Root Setup, Lower-level Setup, Extraction, Signing, and Verification. For Root Setup, the system parameters are supplemented to include a description of the signature space  $\mathcal{S}$ . Lower-level Setup and Extraction are as above.

**Signing:** A signer inputs  $params$ , its private key  $d$ , and  $M \in \mathcal{M}$  and outputs a signature  $S \in \mathcal{S}$ .

**Verification:** A user inputs  $params$ , the ID-tuple of the signer,  $M \in \mathcal{M}$ , and  $S \in \mathcal{S}$  and outputs “valid” or “invalid.”

Signing and verification must also satisfy a consistency constraint, namely when  $d$  is the private key generated by the Extraction algorithm for ID-tuple, then:

$$\forall M \in \mathcal{M} : \text{Verification}(params, \text{ID-tuple}, M, S) = \text{“valid”}$$

where  $S = \text{Signing}(params, d, M)$ .

The security of our HIDE scheme is based on the difficulty of the Bilinear Diffie-Hellman (BDH) Problem. Let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be two cyclic groups of some large prime order  $q$ . We write  $\mathbb{G}_1$  additively and  $\mathbb{G}_2$  multiplicatively. Our HIDE scheme makes use of a “bilinear” pairing.

**Admissible Pairings:** We will call  $\hat{e}$  an *admissible pairing* if  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  is a map with the following properties:

1. Bilinear:  $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$  for all  $Q, R \in \mathbb{G}_1$  and all  $a, b \in \mathbb{Z}$ .
2. Non-degenerate: The map does not send all pairs in  $\mathbb{G}_1 \times \mathbb{G}_1$  to the identity in  $\mathbb{G}_2$ .
3. Computable: There is an efficient algorithm to compute  $\hat{e}(Q, R)$  for any  $Q, R \in \mathbb{G}_1$ .

We will also need the mapping  $\hat{e}$  to be symmetric, i.e.,  $\hat{e}(Q, R) = \hat{e}(R, Q)$  for all  $Q, R \in \mathbb{G}_1$ , but this follows immediately from the bilinearity property and the fact that  $\mathbb{G}_1$  is a cyclic group. We note that the Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such bilinear maps, as in [13.5.7]; see also [14.2].

**BDH Parameter Generator:** As in [5], we say that a randomized algorithm  $\mathcal{IG}$  is a BDH parameter generator if  $\mathcal{IG}$  takes a security parameter  $K > 0$ , runs in time polynomial in  $K$ , and outputs the description of two groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  of the same prime order  $q$  and the description of an admissible pairing  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ .

**Bilinear Diffie-Hellman (BDH) Problem:** Given a randomly chosen  $P \in \mathbb{G}_1$ , as well as  $aP$ ,  $bP$ , and  $cP$  (for unknown randomly chosen  $a, b, c \in \mathbb{Z}/q\mathbb{Z}$ ), compute  $\hat{e}(P, P)^{abc}$ .

For the BDH problem to be hard,  $\mathbb{G}_1$  and  $\mathbb{G}_2$  must be chosen so that there is no known algorithm for efficiently solving the Diffie-Hellman problem in either  $\mathbb{G}_1$  or  $\mathbb{G}_2$ . Note that if the BDH problem is hard for a pairing  $\hat{e}$ , then it follows that  $\hat{e}$  is non-degenerate.

**Bilinear Diffie-Hellman Assumption:** As in [5], if  $\mathcal{IG}$  is a BDH parameter generator, the advantage  $\text{Adv}_{\mathcal{IG}}(\mathcal{B})$  that an algorithm  $\mathcal{B}$  has in solving the BDH problem is defined to be the probability that the algorithm  $\mathcal{B}$  outputs  $\hat{e}(P, P)^{abc}$  on inputs  $\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP$ , where  $(G_1, G_2, \hat{e})$  is the output of  $\mathcal{IG}$  for sufficiently large security parameter  $K$ ,  $P$  is a random generator of  $\mathbb{G}_1$ , and  $a, b, c$  are random elements of  $\mathbb{Z}/q\mathbb{Z}$ . The Bilinear Diffie-Hellman assumption is that  $\text{Adv}_{\mathcal{IG}}(\mathcal{B})$  is negligible for all efficient algorithms  $\mathcal{B}$ .

### 3 Hierarchical ID-Based Encryption Schemes

We describe our scheme in a format similar to that used in [6]. We begin by describing a basic scheme, and then extend it to a full scheme that is secure against adaptive chosen ciphertext attack in the random oracle model, assuming the difficulty of the BDH problem.

We may sometimes refer to elements of  $\mathbb{G}_1$  as “points,” which may suggest that  $\hat{e}$  is a modified Weil or Tate pairing, but we note again that any admissible pairing  $\hat{e}$  will work.

#### 3.1 BasicHIDE

Let  $\text{Level}_i$  be the set of entities at level  $i$ , where  $\text{Level}_0 = \{\text{Root PKG}\}$ . Let  $K$  be the security parameter given to the setup algorithm, and let  $\mathcal{IG}$  be a BDH parameter generator.

*Root Setup:* The root PKG:

1. runs  $\mathcal{IG}$  on input  $K$  to generate groups  $\mathbb{G}_1, \mathbb{G}_2$  of some prime order  $q$  and an admissible pairing  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ ;
2. chooses an arbitrary generator  $P_0 \in \mathbb{G}_1$ ;
3. picks a random  $s_0 \in \mathbb{Z}/q\mathbb{Z}$  and sets  $Q_0 = s_0 P_0$ ;
4. chooses cryptographic hash functions  $H_1: \{0, 1\}^* \rightarrow \mathbb{G}_1$  and  $H_2: \mathbb{G}_2 \rightarrow \{0, 1\}^n$  for some  $n$ . The security analysis will treat  $H_1$  and  $H_2$  as random oracles.

The message space is  $\mathcal{M} = \{0, 1\}^n$ . The ciphertext space is  $\mathcal{C} = \mathbb{G}_1^t \times \{0, 1\}^n$  where  $t$  is the level of the recipient. The system parameters are  $\text{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2)$ . The root PKG’s secret is  $s_0 \in \mathbb{Z}/q\mathbb{Z}$ .

*Lower-Level Setup:* Entity  $E_t \in \text{Level}_t$  picks a random  $s_t \in \mathbb{Z}/q\mathbb{Z}$ , which it keeps secret.

*Extraction:* Let  $E_t$  be an entity in  $\text{Level}_t$  with ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$ , where  $(\text{ID}_1, \dots, \text{ID}_i)$  for  $1 \leq i \leq t$  is the ID-tuple of  $E_t$ ’s ancestor at  $\text{Level}_i$ . Set  $S_0$  to be the identity element of  $\mathbb{G}_1$ . Then  $E_t$ ’s parent:

1. computes  $P_t = H_1(\text{ID}_1, \dots, \text{ID}_t) \in \mathbb{G}_1$ ;
2. sets  $E_t$ ’s secret point  $S_t$  to be  $S_{t-1} + s_{t-1} P_t = \sum_{i=1}^t s_{i-1} P_i$ ;
3. also gives  $E_t$  the values of  $Q_i = s_i P_0$  for  $1 \leq i \leq t-1$ .

*Encryption:* To encrypt  $M \in \mathcal{M}$  with the ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$ , do the following:

1. Compute  $P_i = H_1(\text{ID}_1, \dots, \text{ID}_i) \in \mathbb{G}_1$  for  $1 \leq i \leq t$ .
2. Choose a random  $r \in \mathbb{Z}/q\mathbb{Z}$ .
3. Set the ciphertext to be:

$$C = [rP_0, rP_2, \dots, rP_t, M \oplus H_2(g^r)] \text{ where } g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2.$$

*Decryption:* Let  $C = [U_0, U_2, \dots, U_t, V] \in \mathcal{C}$  be the ciphertext encrypted using the ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$ . To decrypt  $C$ ,  $E_t$  computes:

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^t \hat{e}(Q_{i-1}, U_i)}\right) = M.$$

This concludes the description of our BasicHIDE scheme.

*Remark 1.* Each lower-level PKG – say, in  $\text{Level}_t$  – has a secret  $s_t \in \mathbb{Z}/q\mathbb{Z}$ , just like the root PKG. A lower-level PKG uses this secret to generate a secret point for each of its children, just as the root PKG does. An interesting fact, however, is that lower-level PKGs need not always use the same  $s_t$  for each private key extraction. Rather,  $s_t$  could be generated randomly for each of the PKG’s children.

*Remark 2.*  $H_1$  can be chosen to be an iterated hash function so that, for example,  $P_i$  may be computed as  $H_1(P_{i-1}, \text{ID}_i)$  rather than  $H_1(\text{ID}_1, \dots, \text{ID}_i)$ .

*Remark 3.* In what follows, we may refer to  $S_t$  as  $E_t$ ’s *private point*, and to  $\{Q_i : 1 \leq i < t\}$  as  $E_t$ ’s *Q-values*. We say that  $S'_t$  and  $\{Q'_i : 1 \leq i < t\}$  form a *valid private key* for the point-tuple  $(P_1, \dots, P_t)$  if  $S'_t = s_0 P_1 + \sum_{i=1}^{t-1} s'_i P_{i+1}$  and  $Q'_i = s'_i P_0$  for some  $(s'_1, \dots, s'_{t-1}) \in (\mathbb{Z}/q\mathbb{Z})^{t-1}$ .

*Remark 4.* Note that the same  $g$  can be used for all descendants of  $E_1$ . This value can be precomputed.

### 3.2 FullHIDE: HIDE with Chosen Ciphertext Security

In [6], Fujisaki-Okamoto padding [9] is used to convert a basic IBE scheme to an IBE scheme that is chosen ciphertext secure in the random oracle model. In the same way, BasicHIDE can be converted to FullHIDE, a HIDE scheme that is chosen ciphertext secure in the random oracle model. Next we describe the scheme FullHIDE.

*Setup:* As in the BasicHIDE scheme, but in addition choose hash functions  $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}/q\mathbb{Z}$  and  $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ .

*Extraction:* As in the BasicHIDE scheme.

*Encryption:* To encrypt  $M \in \mathcal{M}$  with the ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$ , do the following:

1. compute  $P_i = H_1(\text{ID}_1, \dots, \text{ID}_i) \in \mathbb{G}_1$  for  $1 \leq i \leq t$ ,
2. choose a random  $\sigma \in \{0, 1\}^n$ ,
3. set  $r = H_3(\sigma, M)$ , and
4. set the ciphertext to be:

$$C = [rP_0, rP_2, \dots, rP_t, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)]$$

where  $g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2$  as before.

*Decryption:* Let  $C = [U_0, U_2, \dots, U_t, V, W] \in \mathcal{C}$  be the ciphertext encrypted using the ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$ . If  $(U_0, U_2, \dots, U_t) \notin \mathbb{G}_1^t$ , reject the ciphertext. To decrypt  $C$ ,  $E_t$  does the following:

1. computes

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^t \hat{e}(Q_{i-1}, U_i)}\right) = \sigma,$$

2. computes  $W \oplus H_4(\sigma) = M$ ,
3. sets  $r = H_3(\sigma, M)$  and tests that  $[U_0, U_2, \dots, U_t, V]$  is a BasicHIDE encryption of  $M$  using  $r$  and  $(\text{ID}_1, \dots, \text{ID}_t)$ . If not, it rejects the ciphertext.
4. outputs  $M$  as the decryption of  $C$ .

Note that  $M$  is encrypted as  $W = M \oplus H_4(\sigma)$ . This can be replaced by  $W = E_{H_4(\sigma)}(M)$  where  $E$  is a semantically secure symmetric encryption scheme (see [9] and Section 4.2 of [6]).

## 4 Hierarchical ID-Based Signature (HIDS) Schemes

ID-based encryption, whether hierarchical or not, has a clear advantage over PKI; it does not require online public key lookup. On the other hand, it is not so clear that ID-based signatures have an advantage over traditional signature schemes using PKI. Indeed, any public-key signature scheme may be transformed into an ID-based (hierarchical) signature scheme by using (a hierarchy of) certificates, since certificates “bind” an identity to a public key.

The previous comments notwithstanding, we present a Hierarchical ID-based Signature (HIDS) scheme based on the difficulty of solving the Diffie-Hellman problem in the group  $\mathbb{G}_1$ . When viewed in isolation, this HIDS scheme is not especially useful for the reasons stated above (though it may be more efficient). However, as will be explained later, the HIDS scheme becomes quite useful when viewed in combination with the HIDE scheme as a complete package.

#### 4.1 A HIDS Scheme

As noted by Moni Naor (see Section 6 of [6]), an IBE scheme can be immediately converted into a public key signature scheme as follows: the signer’s private key is the master key in the IBE scheme. The signer’s signature on  $M$  is the IBE decryption key  $d$  corresponding to the “public key”  $H_1(\text{ID}) = H_1(M)$ . The verifier checks the signature by choosing a random message  $M'$ , encrypting  $M'$  with  $H_1(M)$ , and trying to decrypt the resulting ciphertext with  $d$ . If the ciphertext decrypts correctly, the signature is considered valid.

This observation can be extended to a hierarchical context: a HIDE scheme can be immediately converted to a HIDS scheme. Suppose the signer has ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$ . To sign  $M$ , the signer computes a private key  $d$  for the ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t, M)$ , and sends  $d$  to the verifier. As before, the verifier checks the signature by choosing a random message  $M'$ , encrypting  $M'$  with the “public key”  $(\text{ID}_1, \dots, \text{ID}_t, M)$ , and trying to decrypt the resulting ciphertext with  $d$ . The security of this HIDS scheme follows immediately from the security of our HIDE scheme, since forging a signer’s signature is equivalent to recovering the private key of one of the signer’s children.

An obvious pitfall in the HIDS scheme just described is that an attacker might try to get the signer to sign  $M = \text{ID}_{t+1}$  where  $\text{ID}_{t+1}$  represents an actual identity. In this case, the signer’s signature will actually be a private key, which thereafter may be used to decrypt messages and forge signatures. The easy solution to this problem is to use some expedient – such as a bit prefix – that distinguishes between signing and private key extraction.

Below we describe our HIDS scheme in more detail. The security of the HIDS scheme is based on the difficulty of solving the Diffie-Hellman problem in the group  $\mathbb{G}_1$  (as opposed to HIDE, which requires the BDH problem to be difficult, and therefore requires the Diffie-Hellman problem in  $\mathbb{G}_2$  to be difficult).

Let  $\text{Level}_i$  be the set of entities at level  $i$ , where  $\text{Level}_0 = \{\text{Root PKG}\}$ . Let  $K$  be the security parameter given to the setup algorithm, and let  $\mathcal{IG}$  be a BDH parameter generator.

*Root Setup:* The root PKG:

1. runs  $\mathcal{IG}$  on input  $K$  to generate groups  $\mathbb{G}_1, \mathbb{G}_2$  of prime order  $q$  and an admissible pairing  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ ;
2. chooses an arbitrary generator  $P_0 \in \mathbb{G}_1$ ;
3. picks a random  $s_0 \in \mathbb{Z}/q\mathbb{Z}$  and sets  $Q_0 = s_0 P_0$ ;
4. chooses cryptographic hash functions  $H_1: \{0, 1\}^* \rightarrow \mathbb{G}_1$  and  $H_3: \{0, 1\}^* \rightarrow \mathbb{G}_1$ . The security analysis will treat  $H_1$  and  $H_3$  as random oracles.

The signature space is  $\mathcal{S} = \mathbb{G}_1^{t+1}$  where  $t$  is the level of the signer. The system parameters are  $\text{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_3)$ . The root PKG’s secret is  $s_0 \in \mathbb{Z}/q\mathbb{Z}$ .

*Lower-Level Setup:* As in BasicHIDE.



*Extraction:* As in BasicHIDE.

*Signing:* To sign  $M$  with ID-tuple  $(\text{ID}_1, \dots, \text{ID}_t)$  (using the secret point  $S_t = \sum_{i=1}^t s_{i-1}P_i$  and the points  $Q_i = s_iP_0$  for  $1 \leq i \leq t$ ), do the following:

1. Compute  $P_M = H_3(\text{ID}_1, \dots, \text{ID}_t, M) \in \mathbb{G}_1$ . (As suggested above, we might use a bit-prefix or some other method, instead of using a totally different hash function.)
2. Compute  $\text{Sig}(\text{ID-tuple}, M) = S_t + s_t P_M$ .
3. Send  $\text{Sig}(\text{ID-tuple}, M)$  and  $Q_i = s_i P_0$  for  $1 \leq i \leq t$ .

*Verification:* Let  $[\text{Sig}, Q_1, \dots, Q_t] \in \mathcal{S}$  be the signature for  $(\text{ID-tuple}, M)$ . The verifier confirms that:

$$\hat{e}(P_0, \text{Sig}) = \hat{e}(Q_0, P_1) \hat{e}(Q_t, P_M) \prod_{i=2}^t \hat{e}(Q_{i-1}, P_i).$$

## 5 Shortening the Ciphertext and Signatures

In the HIDE scheme, the length of the ciphertext is proportional to the depth of the recipient in the hierarchy. Similarly, in the hierarchical ID-based signature scheme, the length of the signature is proportional to the depth of the signer in the hierarchy, unless the verifier already has the signer's  $Q$ -values. This section discusses ways in which this ciphertext expansion problem may be avoided.

### 5.1 Dual-HIDE: Dual-Identity-Based Encryption

In 2000, Sakai, Ohgishi and Kasahara [16] presented a “key sharing scheme” based on the Weil pairing. The idea was quite simple: suppose a PKG has a master secret  $s$ , and it issues private keys to users of the form  $sP_y$ , where  $P_y = H_1(\text{ID}_y)$  and  $\text{ID}_y$  is the ID of user  $y$  (as in Boneh-Franklin). Then users  $y$  and  $z$  have a shared secret that only they (and the PKG) may compute, namely,  $\hat{e}(sP_y, P_z) = \hat{e}(P_y, P_z)^s = \hat{e}(P_y, sP_z)$ . They may use this shared secret to encrypt their communications. Notice that this key sharing scheme does not require any interaction between the parties. We can view Sakai, Ohgishi and Kasahara’s discovery as a type of “dual-identity-based encryption,” where the word “dual” indicates that the identities of both the sender and the recipient (rather than just the recipient) are required as input into the encryption and decryption algorithms. The main practical difference between this scheme and the Boneh-Franklin IBE scheme is that the sender must obtain its *private key* from the PKG before sending encrypted communications, as opposed to merely obtaining the *public parameters* of the PKG. For other key agreement schemes that could be viewed as dual-identity-based see [3, 4].

In the hierarchical context, Dual-HIDE may be more efficient than HIDE if the sender and recipient are close to each other in the hierarchy tree. Suppose two users,  $y$  and  $z$ , have the ID-tuples  $(\text{ID}_{y1}, \dots, \text{ID}_{yl}, \dots, \text{ID}_{ym})$  and

$(\text{ID}_{z1}, \dots, \text{ID}_{zl}, \dots, \text{ID}_{zn})$ , where

$$(\text{ID}_{y1}, \dots, \text{ID}_{yl}) = (\text{ID}_{z1}, \dots, \text{ID}_{zl}).$$

In other words, user  $y$  is in  $\text{Level}_m$ , user  $z$  is in  $\text{Level}_n$ , and they share a common ancestor in  $\text{Level}_l$ . User  $y$  may use Dual-HIDE to encrypt a message to user  $z$  as follows:

*Encryption:* To encrypt  $M \in \mathcal{M}$ , user  $y$ :

1. Computes  $P_{zi} = H_1(\text{ID}_{z1}, \dots, \text{ID}_{zi}) \in \mathbb{G}_1$  for  $l+1 \leq i \leq n$ .
2. Chooses a random  $r \in \mathbb{Z}/q\mathbb{Z}$ .
3. Sets the ciphertext to be:

$$C = [rP_0, rP_{z(l+1)}, \dots, rP_{zn}, M \oplus H_2(g_{yl}^r)]$$

where

$$g_{yl} = \frac{\hat{e}(P_0, S_y)}{\prod_{i=l+1}^m \hat{e}(Q_{y(i-1)}, P_{yi})} = \hat{e}(P_0, S_{yl}),$$

$S_y$  is  $y$ 's secret point,  $S_{yl}$  is the secret point of  $y$ 's and  $z$ 's common ancestor at level  $l$ , and  $Q_{yi} = s_{yi}P_0$  where  $s_{yi}$  is the secret number chosen by  $y$ 's ancestor at level  $i$ .

*Decryption:* Let  $C = [U_0, U_{l+1}, \dots, U_n, V]$  be the ciphertext. To decrypt  $C$ , user  $z$  computes:

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_z)}{\prod_{i=l+1}^n \hat{e}(Q_{z(i-1)}, U_i)}\right) = M.$$

Note that if  $y$  and  $z$  have a common ancestor below the root PKG, then the ciphertext is shorter with Dual-HIDE than with non-dual HIDE. Further, using Dual-HIDE, the encrypter  $y$  computes  $m - l + 1$  pairings while the decrypter  $z$  computes  $n - l + 1$  pairings. (Note that  $m + n - 2l$  is the “length” of the path between  $y$  and  $z$  in the hierarchy tree.) In the non-dual HIDE scheme, the encrypter computes one pairing while the decrypter computes  $n$  pairings. Thus when  $m < 2l - 1$ , the total work is less with Dual-HIDE than with non-dual HIDE. The relative computing power of the sender and recipient can also be taken into account. In the full paper we will show how to decrease the number of pairings that  $y$  and  $z$  must compute to  $m + n - 2l + 1$  if their common ancestor in  $\text{Level}_l$  always uses the same  $s_l$  rather than generating this number randomly with each private key extraction.

Dual-HIDE also makes domain-specific broadcast encryption possible. Suppose user  $y$  wants to encrypt a message to everyone having the same ancestor in  $\text{Level}_l$ . Everyone in this common ancestor's domain may compute the shared secret  $\hat{e}(P_0, S_{yl})$ , and so this secret may be used as a shared key of everyone in this domain. Users outside of this domain, other than the parent of the common ancestor, will be unable to compute this pairing. (In Section 6.1, we describe how to exclude even the parent.) Note that Dual-HIDE broadcast is not fully compatible with the HIDS scheme. If Dual-HIDE broadcast and the HIDS scheme

use the same parameters, everyone outside the domain who receives a signature from someone in the domain will also be able to compute  $\hat{e}(P_0, S_{yl})$ .

Fujisaki-Okamoto padding turns Dual-HIDE into FullDual-HIDE, a dual-identity encryption scheme with adaptive chosen ciphertext security.

## 5.2 Dual-HIDS: Dual-Identity-Based Signatures

Dual hierarchical identity-based signatures (Dual-HIDS) are much easier to explain. If users  $y$  and  $z$ , as above, have a common ancestor in  $\text{Level}_l$ , then  $y$  only needs to send  $Q_{yi}$  for  $l + 1 \leq i \leq m$ . This makes the length of the signature proportional to  $m - l$  rather than  $m$ .

## 5.3 Authenticated Lower-Level Root PKGs

Suppose that user  $y$  often sends mail to people at a certain university – say, Cryptography State University (CSU) – but that CSU is deep in the hierarchy, and that  $y$  is not close to CSU in the hierarchy. How do we solve the ciphertext expansion problem? One solution, of course, is for CSU to set up its own root PKG with its own system parameters, unassociated with the “actual” root PKG. After  $y$  obtains CSU’s system parameters, its ciphertext to CSU recipients will be shorter. However, we would prefer not to have “rogue” root PKGs.

A better solution is for CSU to set up a root PKG that is “authenticated” by the actual root PKG. For this purpose, the actual root PKG may have an additional parameter, a random message  $M'$ . To set up its authenticated root PKG, CSU “signs”  $M'$ , generating the signature  $Sig = S_t + s_t P_{M'}$ , where  $S_t$  is CSU’s private point, and  $s_t$  is its lower-level secret. CSU also publishes  $Q_i$  for  $1 \leq i \leq t$ .

Let  $(ID_1, \dots, ID_t, \dots, ID_v)$  be the ID-tuple of user  $z$  at CSU having point-tuple  $(P_1, \dots, P_t, \dots, P_v)$ . Then  $y$  may send an encrypted message to  $z$ , using the parameters for CSU’s authenticated root PKG, as follows:

*Encryption:* To encrypt  $M \in \mathcal{M}$ , user  $y$ :

1. Computes  $P_i = H_1(ID_1, \dots, ID_i) \in \mathbb{G}_1$  for  $t + 1 \leq i \leq v$ .
2. Chooses a random  $r \in \mathbb{Z}/q\mathbb{Z}$ .
3. Sets the ciphertext to be:

$$C = [rP_0, rP_{t+1}, \dots, rP_v, M \oplus H_2(g_t^r)]$$

where

$$g_t = \frac{\hat{e}(P_0, Sig)}{\hat{e}(s_t P_0, P_{M'})} = \hat{e}(P_0, S_t) .$$

*Decryption:* Let  $C = [U_0, U_{t+1}, \dots, U_v, V]$  be the ciphertext. To decrypt  $C$ , user  $z$  computes:

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_v)}{\prod_{i=t+1}^v \hat{e}(Q_{i-1}, U_i)}\right) = M$$

where  $S_v$  is  $z$ ’s private key.

The number of pairings computed by the decrypter is  $v - t + 1$ , one more than its depth below CSU, not its depth below the actual root PKG.

Interestingly, if  $y$  obtains *any* signature from CSU, not necessarily on a particular message  $M'$ , then  $y$  may use that signature to shorten its ciphertext in the same way. In effect,  $y$ 's possession of any signature from CSU allows  $y$  to use Dual-HIDE as if  $y$ 's position in the hierarchy is just below CSU. Thus,  $y$  may use CSU's signature to shorten its ciphertext not only to entities below CSU in the hierarchy, but also to any entity that is *close* to CSU in the hierarchy. In general, one could have an "optimized" HIDE scheme in which the sender stores a list of HIDS signatures that it has obtained, and, upon each encryption, searches through that list (which may be put in lexicographic order) to find the signer that is closest in the hierarchy to the intended message recipient, and then uses that signer's signature, in combination with Dual-HIDE, to minimize the length of the ciphertext.

## 6 Restricting Key Escrow

In IBE schemes, key escrow is "inherent" because the PKG knows the private key of each user. Even in the hierarchical scheme of Horwitz and Lynn, every ancestor of a given user in the hierarchy knows that user's private key. Although this key escrow property may be useful in some contexts, it is certainly not desirable for all applications.

In our HIDE scheme, since the private point of a user depends on a secret number known only to the parent of that user, no ancestor other than the parent may compute the user's particular private point. However, the user's ancestors can still decrypt the user's mail; they may simply compute a different (but equally effective) private key for the user based on different lower-level  $Q$ -values. Using these different  $Q$ -values, they may also forge the user's signature. In this section, we discuss how Dual-HIDE and/or key agreement protocols can be used to restrict this key escrow property.

### 6.1 Restricting Key Escrow Using Dual-HIDE

Consider again users  $y$  and  $z$  from Section 5.1 who have a common ancestor in  $\text{Level}_l$ . Let's say their common ancestor is Cryptography State University, and suppose that user  $y$  uses Dual-HIDE to encrypt its messages to  $z$ . As stated above, CSU's parent knows CSU's private point. From CSU's perspective, this may be an undesirable situation. However, CSU can easily change its private point  $S_l$  by setting  $S_l := S_l + bP_l$  and setting  $Q_{l-1} := Q_{l-1} + bP_0$  for some random  $b \in \mathbb{Z}/q\mathbb{Z}$ . This new private key is just as effective, and is unknown to CSU's parent. Assuming that CSU uses its new private key to issue private keys to its children, none of CSU's ancestors will be able to decrypt  $y$ 's message to  $z$  encrypted using Dual-HIDE. More specifically, only ancestors of  $z$  that are within CSU's domain will be able to decrypt.

## 6.2 Authenticated Key Agreement with no Session Key Escrow

HIDS provides a convenient platform on which key agreement may be authenticated (see also [1] for authenticated three-party (non-ID based) key agreement protocols using pairings). A simple explicit authenticated key agreement protocol is as follows:

1. Alice chooses a random  $a \in \mathbb{Z}/q\mathbb{Z}$  and sends  $aP_0$  and  $Sign(aP_0)$  to Bob.
2. Bob chooses a random  $b \in \mathbb{Z}/q\mathbb{Z}$  and sends  $bP_0$  and  $Sign(bP_0)$  to Bob.
3. Alice and Bob verify the received signatures and compute the shared secret:  $abP_0$ .

Here, there is no session key escrow. However, there is still an attack scenario: an ancestor of Alice and an ancestor of Bob could collude to mount a man-in-the-middle attack. This attack has an analogue in PKI: CAs could collude in a similar way. Dual-HIDE can be used in combination with key agreement to minimize the possible scope of such collusion among ancestors.

Implicit authentication based on Sakai-Ohgishi-Kasahara key agreement can be done as follows. Alice and Bob first perform a standard (or elliptic curve) Diffie-Hellman exchange, after which Alice thinks the shared Diffie-Hellman value is  $g_A$  and Bob thinks it is  $g_B$ . Then Alice computes the shared secret as  $H(g_A, S_{AB})$  and Bob computes it as  $H(g_B, S_{AB})$ , where  $H$  is a one-way collision-resistant hash function and  $S_{AB} = \hat{e}(P_A, P_B)^s = \hat{e}(S_A, P_B) = \hat{e}(S_B, P_A)$ , where  $P_A = H_1(\text{ID}_A)$  is Alice's public point and  $S_A = sP_A$  is her private point,  $P_B = H_1(\text{ID}_B)$  is Bob's public point and  $S_B = sP_B$  is Bob's private point, and  $s$  is their PKG's master secret. Unless the man-in-the-middle is the PKG, it will not be able to compute Alice's or Bob's version of the shared secret, since it does not know  $S_{AB}$ . However, it can prevent Alice and Bob from computing the same shared secret. Alice and Bob will not know that their key agreement protocol has been disrupted until, for example, one sends an undecipherable message to the other. A passive PKG will not know Alice's and Bob's shared Diffie-Hellman value, and is therefore unable to compute the session key.

## 7 Security

### 7.1 Security Definitions

We first give some definitions that are very similar to those given in [56,12]. Their similarity should not be surprising because, at a high level, the security issues involved in hierarchical ID-based cryptography are substantially identical to those in non-hierarchical ID-based cryptography; we are merely adding new levels.

**Chosen-Ciphertext Security:** As Boneh and Franklin noted in the context of (non-hierarchical) ID-based cryptography, the standard definition of chosen-ciphertext security must be strengthened for ID-based systems, since one should

assume that an adversary can obtain the private key associated with any identity of its choice (other than the particular identity being attacked). The same applies to hierarchical ID-based cryptography. Thus, we allow an attacker to make “private key extraction queries.” Also, as in [6], we allow the adversary to choose the identity on which it wishes to be challenged.

One subtlety is that an adversary may choose its target identity adaptively or nonadaptively. An adversary that chooses its target adaptively will first make hash queries and extraction queries, and then choose its target based on the results of these queries. Such an adversary might not have a particular target in mind when it begins the attack, and its eventual target need not even belong to an existing entity. Rather, this adversary is successful if it is able to hack some identity to which it is not entitled. A nonadaptive adversary, on the other hand, chooses its target independently from results of hash queries and extraction queries. For example, such an adversary might target a personal enemy. The adversary may still make hash and extraction queries, but its target choice is based strictly on the target’s identity, not on query results. Obviously, security against an adaptively-chosen-target adversary is the stronger, and therefore preferable, notion of security. However, we will address both types of security, since our security proofs against nonadaptively-chosen-target adversaries are stronger.

We say that a HIDE scheme is semantically secure against adaptive chosen ciphertext and adaptive (resp., nonadaptive) chosen target attack (IND-HID-CCA (resp. IND-NHID-CCA)) if no polynomially bounded adversary  $\mathcal{A}$  has a non-negligible advantage against the challenger in the following game. (Note: for IND-NHID-CCA, Phase 1 is omitted.)

**Setup:** The challenger takes a security parameter  $K$  and runs the Root Setup algorithm. It gives the adversary the resulting system parameters  $params$ . It keeps the root key to itself.

**Phase 1:** The adversary issues queries  $q_1, \dots, q_m$  where  $q_i$  is one of:

1. Public-key query (ID-tuple $_i$ ): The challenger runs a hash algorithm on ID-tuple $_i$  to obtain the public key  $H(\text{ID-tuple}_i)$  corresponding to ID-tuple $_i$ .
2. Extraction query (ID-tuple $_i$ ): The challenger runs the Extraction algorithm to generate the private key  $d_i$  corresponding to ID-tuple $_i$ , and sends  $d_i$  to the adversary.
3. Decryption query (ID-tuple $_i, C_i$ ): The challenger runs the Extraction algorithm to generate the private key  $d_i$  corresponding to ID-tuple $_i$ , runs the Decryption algorithm to decrypt  $C_i$  using  $d_i$ , and sends the resulting plaintext to the adversary.

These queries may be asked adaptively. Note also that the queried ID-tuple $_i$  may correspond to a position at any level in the hierarchy.

**Challenge:** Once the adversary decides that Phase 1 is over, it outputs two equal length plaintexts  $M_0, M_1 \in \mathcal{M}$  and an ID-tuple on which it wishes to be challenged. The only constraints are that neither this ID-tuple nor its ancestors

appear in any private key extraction query in Phase 1. Again, this ID-tuple may correspond to a position at any level in the hierarchy. The challenger picks a random bit  $b \in \{0, 1\}$  and sets  $C = \text{Encryption}(params, \text{ID-tuple}, M_b)$ . It sends  $C$  as a challenge to the adversary.

**Phase 2:** The adversary issues more queries  $q_{m+1}, \dots, q_n$  where  $q_i$  is one of:

1. Public-key query ( $\text{ID-tuple}_i$ ): Challenger responds as in Phase 1.
2. Extraction query ( $\text{ID-tuple}_i \neq \text{ID-tuple}$  or ancestor): Challenger responds as in Phase 1.
3. Decryption query ( $(\text{ID-tuple}_i, C_i) \neq (\text{ID-tuple}, C)$ ): Challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess  $b' \in \{0, 1\}$ . The adversary wins the game if  $b = b'$ . We define its advantage in attacking the scheme to be  $|\Pr[b = b'] - \frac{1}{2}|$ .

**One Way Identity-Based Encryption:** As in [5], we define one-way encryption (OWE) for a public key encryption scheme as follows. The adversary  $\mathcal{A}$  is given a random public key  $K_{pub}$  and a ciphertext  $C$  that is the encryption of a random message  $M$  using  $K_{pub}$ , and outputs a guess for the plaintext. The adversary is said to have advantage  $\epsilon$  against the scheme if  $\epsilon$  is the probability that  $\mathcal{A}$  outputs  $M$ . The scheme is said to be a one-way encryption (OWE) scheme if no polynomial time adversary has a non-negligible advantage in attacking the scheme.

We say that a HIDE scheme is one-way (HID-OWE or NHID-OWE, depending on whether the target is chosen adaptively or not) if no polynomial time adversary has a non-negligible advantage against the challenger in the following game. (Phase 1 is omitted for NHID-OWE.)

**Setup:** The challenger takes a security parameter  $k$  and runs the Root Setup algorithm. It gives the adversary the resulting system parameters  $params$ . It keeps the root key to itself.

**Phase 1:** The adversary makes public-key and/or extraction queries as in Phase 1 above.

**Challenge:** Once the adversary decides that Phase 1 is over, it outputs a new ID-tuple on which it wishes to be challenged. The challenger picks a random  $M \in \mathcal{M}$  and sets  $C = \text{Encryption}(params, \text{ID-tuple}, M)$ . It sends  $C$  as a challenge to the adversary.

**Phase 2:** The adversary issues more public-key queries and more extraction queries on identities other than this ID-tuple and its ancestors, and the challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess  $M' \in \mathcal{M}$ . The adversary wins the game if  $M = M'$ . We define the adversary's advantage in attacking the scheme to be  $\Pr[M = M']$ .

## 7.2 Security Results

The security of BasicHIDE and Dual-HIDE is based on the difficulty of the BDH problem, as stated in the following theorems (which are analogous to Theorem 4.1 in [6]):

**Theorem 1.** *Suppose there is an NHID-OWE adversary  $\mathcal{A}$  that has advantage  $\epsilon$  against the BasicHIDE or Dual-HIDE scheme for some ID-tuple and that makes  $q_{H_2} > 0$  hash queries to the hash function  $H_2$  and a finite number of private key extraction queries. If the hash functions  $H_1, H_2$  are random oracles, then there is an algorithm  $\mathcal{B}$  that solves the BDH in groups generated by  $\mathcal{IG}$  with advantage at least  $(\epsilon - \frac{1}{2^n})/q_{H_2}$  and running time  $O(\text{time}(\mathcal{A}))$ .*

**Theorem 2.** *Suppose there is an HID-OWE adversary  $\mathcal{A}$  that makes at most  $q_{H_2} > 0$  hash queries to the hash function  $H_2$  and at most  $q_E > 0$  private key extraction queries and has advantage  $\epsilon_t$  of successfully targeting a BasicHIDE or Dual-HIDE node in  $\text{Level}_t$ . If the hash functions  $H_1, H_2$  are random oracles, then there is an algorithm  $\mathcal{B}$  that solves the BDH in groups generated by  $\mathcal{IG}$  with advantage at least  $(\epsilon_t(\frac{t}{e(q_E+t)})^t - \frac{1}{2^n})q_{H_2}^{-1}$  and running time  $O(\text{time}(\mathcal{A}))$ .*

If  $t = O(1)$  and  $q_E$  is polynomial in the security parameter, then  $(t/e(q_E+t))^t$  is non-negligible in the security parameter, and we have a polynomial reduction from BasicPub to BasicHIDE (for adaptively-chosen-target adversaries).

With Fujisaki-Okamoto padding, these schemes can be made chosen ciphertext secure if BDH is hard in the groups generated by  $\mathcal{IG}$ . The proof follows from Theorems 1 and 2 analogously to the way that Theorem 4.4 of [6] follows from Lemma 4.3 of [6]. Further, the security of the HIDS scheme depends only on the difficulty of the Diffie-Hellman problem in the group  $\mathbb{G}_1$ , and not on BDH. We will give security proofs in the full version of the paper.

## 8 Extensions and Observations

**Improving Efficiency of Encryption:** Levels 0 and 1 can be merged into a single (combined levels 0 and 1) root PKG. In that case,  $g = \hat{e}(Q_0, P_1)$  is included in the system parameters. This saves encrypters the task of computing the value of this pairing. However, decrypters must compute an extra pairing (as a result of being one level lower down the tree).

**Distributed PKGs:** As in Section 6 of [6], the secrets  $s_i$  and private keys can be distributed using techniques of threshold cryptography to protect the secrets and make the scheme robust against dishonest PKGs.

**Concrete Schemes:** For our HIDE and HIDS schemes, one can use the same elliptic curves or abelian varieties as those in [6], [7], or [15].



## 9 Conclusion

We gave hierarchical ID-based encryption (HIDE) schemes that are practical, totally collusion-resistant, and secure against chosen-ciphertext attacks. The message expansion factor and complexity of decryption grow only linearly with the number of levels in the hierarchy. We introduced a related hierarchical ID-based signature (HIDS) scheme that is especially effective when used in combination with HIDE and Dual-HIDE. This also appears to be the first paper related to ID-based cryptography that gives methods for circumventing key escrow.

## Acknowledgments

We thank Jonathan Katz, Yiqun Lisa Yin, James Kempf, Anand Desai, and Satomi Okazaki for helpful conversations.

## References

1. S. S. Al-Riyami and K. G. Paterson. Authenticated three party key agreement protocols from pairings. Cryptology e-Print Archive, <http://eprint.iacr.org/2002/035/>.
2. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems, in *Advances in Cryptology – Crypto 2002, Lecture Notes in Computer Science* **2442** (2002), Springer, 354–368.
3. R. Blom. An optimal class of symmetric key generation systems, in *Advances in Cryptology – Eurocrypt ’84, Lecture Notes in Computer Science* **209** (1984), Springer, 335–338.
4. C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung. Perfectly-secure key distribution for dynamic conferences, in *Advances in Cryptology – Crypto ’92, Lecture Notes in Computer Science* **740** (1993), Springer, 471–486.
5. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing, in *Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science* **2139** (2001), Springer, 213–229.
6. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing, extended version of [5], <http://www.cs.stanford.edu/~dabo/papers/ibe.pdf>.
7. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing, in *Advances in Cryptology – Asiacrypt 2001, Lecture Notes in Computer Science* **2248** (2001), Springer, 514–532.
8. C. Cocks. An identity based encryption scheme based on quadratic residues, in *Cryptography and Coding, Lecture Notes in Computer Science* **2260** (2001), Springer, 360–363.
9. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes, in *Advances in Cryptology – Crypto ’99, Lecture Notes in Computer Science* **1666** (1999), Springer, 537–554.
10. G. Hanaoka, T. Nishioaka, Y. Zheng, and H. Imai. An efficient hierarchical identity-based key-sharing method resistant against collusion-attacks, in *Advances in Cryptology – Asiacrypt 1999, Lecture Notes in Computer Science* **1716** (1999), Springer, 348–362.

11. G. Hanaoka, T. Nishioaka, Y. Zheng, and H. Imai, A hierarchical non-interactive key-sharing scheme with low memory size and high resistance against collusion attacks, to appear in *The Computer Journal*.
12. J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption, in Advances in Cryptology – Eurocrypt 2002, Lecture Notes in Computer Science **2332** (2002), Springer, 466–481.
13. A. Joux. A one round protocol for tripartite Diffie-Hellman, in Algorithmic Number Theory (ANTS-IV), Lecture Notes in Computer Science **1838** (2000), Springer, 385–394.
14. V. Miller. Short programs for functions on curves, unpublished manuscript.
15. K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology, in Advances in Cryptology – Crypto 2002, Lecture Notes in Computer Science **2442** (2002), Springer, 336–353.
16. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. SCIC 2000-C20, Okinawa, Japan, January 2000.
17. A. Shamir. Identity-based cryptosystems and signature schemes, in Advances in Cryptology – Crypto '84, Lecture Notes in Computer Science **196** (1984), Springer, 47–53.

# Crypto-integrity

Moti Yung

Department of Computer Science, Columbia University,  
`moti@cs.columbia.edu`

**Abstract.** Designing cryptographic mechanisms and products is a challenging task. This task will become increasingly hard as software technology and systems evolve and as the new computational environment becomes more distributed, more diverse, and more global. In order to enable the inclusion of cryptographic components in the future infrastructure and within future applications, it is argued that assurance of their (secure) operation has to be provided and their robustness has to be exhibited in real time. This assurance, which we call *crypto-integrity* will guarantee the correct functioning of the cryptographic components in an efficient fashion. This built-in integrity should have no impact on the system security and should have minimal impact on its function, performance and composability.

We review the need for crypto-integrity in various known settings, ways to implement it based on known protocol techniques as well as potential future directions. The paper is written as a position paper and not as a survey of the vast relevant literature.

## 1 Introduction

Integrity assurance is a part of many modern cryptography constructions. In fact, cryptography itself is often employed to provide strong integrity such as “message integrity” (assured by hashing a message based on a secret key in a MAC operation). Other cryptographic operations have integrity associated with them, e.g. digital signing (initiated by Diffie-Hellman, Rivest Shamir and Adleman, and Rabin) involves a verification procedure which assures the authenticity of the signature.

The design of cryptographic protocols where many parties are involved in a joint activity allows dishonest adversaries to behave in arbitrary devious ways. Thus, the need to assure well behaved parties arises naturally. Early protocols like Rabin’s signature protocol and Blum’s coin flipping had assurance of behavior designed into them. Then, the development of the basic notion of zero-knowledge by Goldwasser, Micali and Rackoff was crucial to recognizing the central idea of systematic assurance of well behaved parties. The fact that NP languages have zero-knowledge proofs (and arguments) is fundamental and can be used to assure that actions taken in a cryptographic protocol are in accordance with the protocol specifications. This serves as a general plausibility result that integrity of parties in a protocol can be upheld and monitored.

However, for every protocol solving a specific task, we need to design specific proofs and integrity mechanisms that are efficient and suitable to its setting. In fact, as cryptographic services are deployed, every system configuration and every specialized setting will need to provide efficient and specialized methods for exhibiting the correct behavior of components; we call these methods *crypto-integrity*. Next we mention some systems factors affecting the need for specialized crypto-integrity.

### 1.1 System Setting and Requirements

We will now review some practical requirements and will argue how they can be achieved/ strengthened by having crypto-integrity functions.

As cryptographic primitives and protocols are being developed as products, their adoption into the computing and communication infrastructure will be based on their usefulness and effectiveness (typically measured in business by “Return on Investment”). This means that certain basic properties are required, some of which are related to generic desired properties (like user-friendliness), while others relate directly to the intrinsic properties of cryptographic design:

- Generality and composability of components: the basic product should be useful in many settings as a general primitive. We should be able to employ it with many (current and future) applications and it should remain secure in these settings.
- Adaptability (or scalable security): we should be able to embed the product in various settings (of different scales) and change its environment and even the threats to which it is susceptible, yet it should keep on being secure. (Many designs are too “setting specific” and are hard to adapt to new environments).
- Performance: this is an important factor that may fail the product when e.g. speed is a requirement or when it becomes too costly to implement fast or compact solutions. In some environments performance criteria are crucial (and this changes as technology changes).
- Assurance: there should be assurance about the product workings. Besides the proof of security which should be done in the proper setting of the entire application (end-to-end arguments), and besides testing, it will be useful if the product will have on-line assurance of the way its components work (namely, what we called crypto-integrity).

The above and similar requirements usually serve as a feedback to the crucial work on foundations of cryptography, where new notions are defined, designed and improved, and where the characteristic and inherent properties of the basic notions are investigated.

These requirements are also very useful to practitioners. To have a sustainable business one needs to have certain quality in its products. Having crypto-integrity may ease re-usability of components and shorten the test cycle. Having a general component that is adaptable to various settings and can support current and emerging applications is, at times, an important prerequisite for profit.

The notion of crypto-integrity has implications to all the above requirements. It helps assure the proper behavior of components, which means that parties are committed to certain computations, something that leads to predictable performance (and time-out mechanisms can further detect delays that are system specific methods to cope with delays caused by misbehaving components). It helps in exhibiting (at the interface between components) what is done by individual components and sub-systems, and thus helps in composing systems and the adaptability of components. It supports and reinforces formal assurance procedures such as certification of products by government bodies.

Crypto-integrity is an assurance mechanism which is achieved by enhancing the function of the component itself (in on-line operation). On-line integrity has many implications in special contexts. Let us mention one current implication. The context is a trusted computing environment run under a tamper resistant component of the architecture. This setting may have a lot of positive implications. It has however, many bad implications, if it is not run according to a “publicly agreed upon” specification. With crypto-integrity we may be able, at least partially, to assure compliance of a well specified trusted environment with its global specification (especially if we limit it to very specialized functions, since in general we cannot really tell what a tamper resistance cryptographic environment is doing as was shown by the notion of Kleptography [Young and Yung, Crypto 96, Eurocrypt 97, Crypto 97]).

In the rest of this paper I will mention examples of mechanisms (protocol design and settings) where crypto-integrity plays an important role.

## 2 Examples: the Usefulness of Crypto-integrity

### 2.1 Cryptographic Program Checking

Blum introduced the useful and elegant notion of *Program Result Checking*. In this setting, given arbitrary input  $\alpha$  and program  $P$ , a checker  $C$  for a function  $f$  will catch, with high probability, if  $P(\alpha) \neq f(\alpha)$ . The checker has only “black-box” access to the program and accomplishes its goal on-line. Cryptographic program checking (developed in [Frankel, Gemmel and Yung, STOC96]) allows the on-line checking of programs computing cryptographic functions in a working environment.

In this model the checker worries about correctness (a concern that traditional “program checking” takes care of), since due to the adversarial setting we require correctness with very high probability. In addition, the owner of a program will output  $P(\alpha)$  provided it is authorized to output the result, but the checker (user) learns nothing more about  $P$  from this checking procedure, in the spirit of the zero-knowledge complexity approach to knowledge. Such checking methods are witness-based (they allow the output of a few values to be known as a witness) and achieve fast verification. In some sense the procedures can be viewed as extending the “deniable signature” proof method of Chaum.

The basic application of this method is testing cryptographic servers. In the future, many servers will act on behalf of user populations and assurance of

non-spoofed service will be important. We now discuss several applications for cryptographic program checking.

Consider the *encrypting-machine requester game* where the encrypting-machine (server) is willing to encrypt authorized requests. If the checking process requires the encryption of other (unauthorized) plaintext so that the output of the request can be checked, then the checker can exploit this service to encrypt unauthorized texts as well. Cryptographic program checking can be used to prevent such an exposure.

Another similar application is the *international key escrow game*. This is related to specifications of key recovery methods between organizations and entities, an issue that is not yet well understood (on a technical level), but is similar to the concept of escrow encryption systems.

In this situation country  $A$  has a key escrow system and will allow country  $B$  to obtain decryption of messages of  $A$ 's citizens under some predefined treaty and under some conditions.  $A$  does not want to provide  $B$  with the actual keys of its citizens (only the decrypted messages should be disclosed) while country  $B$  does not trust that  $A$  will reply with the correct cleartext values. Cryptographic program checking, in turn, allows  $B$  to verify the correctness of the outputs, while  $A$  knows that it is not being abused (by revealing messages not covered by the treaty or conditions). Of course, the setting is applicable to many (less controversial) scenarios. The basic ideas of our methodology can be applied to a *verifier hardware-device game*, where a holder of a result computed by some hardware device needs to probe a verifying device. For example, it makes it possible to make sure that a value computed in the past (a time stamp) is correct without the verification process leaking the computation itself (thus, recomputing the time stamp— which in effect causes an undesirable back-stamping). The methodology of cryptographic program checking applies to this situation as well.

## 2.2 Threshold Cryptosystems

One of the applications that motivated this research on cryptographic program checking is in the development of verification algorithms for threshold cryptography (where a function sharing or capability sharing is taking place). This is a method to distribute control of a function by a dealer or distributedly. In the *function sharing game* a function  $f$  is distributed amongst  $n$  agents as programs  $P_1(\cdot), \dots, P_n(\cdot)$  such that a threshold (quorum) of, say, any  $t$  are able to compute  $f(\alpha)$  from  $P_{i_1}(\alpha), \dots, P_{i_t}(\alpha)$ . There are several interesting applications for which function sharing is a very useful solution in practice (e.g., distributed decryption, signature generation, public key certification generation, e-cash generation, etc.).

Once the shares are available there is a polynomial-time combiner that collects the shares and combines them to the final result of the function. When agents misbehave this gives rise to the game between the agents and the combiner, where the combiner has to be sure to pick correct shares into its computation. If there is no efficient way to verify correctness of shares, the combiner may need to try all subsets of shares (but this will take exponential time). The

agent combiner game will assure the combiner which of the agents acted correctly. The need for *robust* function sharing was also expressed in an application for replicating services in a network where some of the clients and servers have been corrupted by an adversary. Since there is a real systems' need for the primitive, inefficient methods like non-interactive zero-knowledge techniques should be avoided. While many results have been achieved in this area, the applicability and usability of the results has yet to be realized.

## 2.3 Proactive Security

Another game, called the *proactive function sharing game*, is an extension of the robust function sharing game. In this system the agents' state is modified over time (by the agents themselves) so that an adversary may have access to all agents over the lifetime of the system but not to all (or not even to a quorum) at any particular point in time. The agents periodically modify their state so that information learned by a mobile adversary at two points in time is, for practical purposes, uncorrelated. In this game, the honest agents make sure that changing their state does not provide a means for the adversary to learn too much information nor to destroy the ability of honest agents to later compute the function. Hence each of the agents verifies that the information provided to it by other agents is correct before it changes states. This notion is called proactive security.

The notion of proactive security was a result of dealing with a mobile adversary which corrupts different parts of the system in different times. It was motivated by new threats like network viruses. It was first developed for the area of general secure multi-party computation (initiated by Yao and Goldreich, Micali and Wigderson) in [Ostrovsky and Yung, PODC 91]. It was somewhat motivated by an early notion of allowing users in this setting of general multi-party computation to leave and re-join the computation smoothly. (This last idea needed the method of "share-of-shares" which is a robustness mechanism was first employed in a work [Galil, Haber and Yung, Crypto 87]). It was also further motivated by Dijkstra's notion of self-stabilizing protocols which allows transient faults, whereas proactive protocols allows persisting faults (rather than transient) by introducing redundancy (requiring honest majority).

Many procedures have been "proactivized" and in particular so have many distributed cryptosystems. The need for integrity when we have the system dynamically changing and when honest users re-join, is crucial.

Proactive methods allow us to change the quorum of users that hold some computational capability distributedly within a system. This is a new function that is made possible by the built-in integrity mechanism which ensures the correctness of the shared capability, throughout.

## 2.4 Voting Schemes

Voting schemes have interesting requirements. They ask that the voter's action is universally verifiable yet his ballot has to remain secret. Various methods

assuring the integrity of the system and limiting malicious voters, preventing them from disturbing the global voting process have been developed. Recently (in [Kiayias and Yung, PKC 2002]) a small scale election with unique properties was given. It assures increased privacy (where in order to compromise the privacy of a ballot, all other voters have to collude against an individual) and combines it with a form of fault tolerance and universal verifiability in a way that there are no disputes in the on-line process (dispute freeness) due to the built-in crypto-integrity.

## 2.5 Assurance with respect to Off-line Third Parties

In escrow and key recovery systems (especially in auto recoverable cryptosystems, see [Young and Yung, Eurocrypt'98, PKC'00]), as well as in traceable e-cash and in various other settings, we have a user assuring that some action by an off-line third party is in fact doable, once this third party becomes active. The availability of public keys and the proof techniques which use them, enable such proof of actions by a third party where there is no need for on-line parties to participate. we expect such methods to find further applications in many areas.

## 2.6 Minimizing Key-Exposure

Recently, cryptosystems have been designed where key exposure is coped with either by “forwarding” (self updating) the key, making past keys inaccessible, or by sharing the key with a server (key-insulated cryptosystems developed in [Dodis, Katz, Xu and Yung, Eurocrypt'02]). We note that when sharing and updating keys with other elements, crypto-integrity is a must.

## 2.7 Multi User Setting

The case where multi users are present in a system (rather than two parties: a sender and a receiver) may change the setting and the possibility of procedures that can be employed. For example a “distributed proof” can be conducted assuring a group of users with honest majority of a fact while not revealing the underlying secret. Many areas of multi-user oriented cryptography are still open, and crypto-integrity is likely to play an increasingly important role in this setting.

## 2.8 Environmental Constraints and Protocols

Due to technological changes, the environment where protocols are being executed is changing. Protocol notions based on the Internet concurrency, and notions based on limited execution environments (mobile devices and smart cards) are being considered nowadays. These changes will affect the crypto-integrity requirements among other changes that they will dictate.

Environmental constraints also motivate research on modularity and composition of crypto protocols. The methods that assure integrity are paramount to



allowing properties like composition in specialized settings. For example, the issue of self-testing protocols while retaining the protocol's security is in its infancy (see [Franklin, Garay and Yung, DISC'99]).

### 3 Conclusions

We reviewed areas where crypto-integrity methods have been developed and used extensively. We showed how the notion allows for integrity assurance while retaining the secrecy of cryptographic techniques.

We claim that on-line assurance is a crucial security component: If we run tests of the systems in a working environment where we have to give up security to validate the system's correctness, we are at risk that someone in control of moving from test mode to operational mode can use this capability to compromise the system.

Also, on-line assurance is very important in working systems which evolve and change. It makes sure the core cryptographic component is acting correctly. Maintaining the integrity as the system changes is an interesting open area of research.

On-line crypto-integrity adds "function" by allowing parties to be off-line but nevertheless assuring that when they join they will be able to perform their task. With cryptography being part of many applications, such tools are expected to be crucial.

The research questions regarding on-line integrity in cryptographic settings are many: from improving the efficiency and other properties of existing methods, through questions related to new techniques and new primitives where integrity is crucial, to possibly new areas where crypto-integrity functionality is a must such as "safe cryptographic testing and development," "general notions of composability and modularity," "theory of reusable cryptographic methods," and "theory of update of system based on change of threats." We believe that given that "cryptosystems deployment as part of more general computing systems" is still (in spite of deployment successes) an area in its infancy, the area of assuring integrity in cryptographic settings is open to further investigation and to innovations.

# Gummy and Conductive Silicone Rubber Fingers

## Importance of Vulnerability Analysis

Tsutomu Matsumoto

Yokohama National University,  
Graduate School of Environment and Information Sciences,  
79-7 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan,  
tsutomu@mlab.jks.ynu.ac.jp

**Abstract.** Vulnerability evaluation of various biometric systems should be conducted and its results should be available to potential users.

### Summary

*Biometrics* is utilized in individual authentication techniques which identify individuals by checking physiological or behavioral characteristics, such as fingerprints, faces, voice, iris patterns, signatures, etc. Biometric systems are said to be convenient because they need neither something to memorize such as passwords nor something to carry about such as ID tokens [1]. In spite of that, a user of biometric systems would get into a dangerous situation when her/his biometric data are abused. For example, you cannot change your fingerprints while you can change your passwords or ID tokens when they are compromised. Therefore, biometric systems must protect the information for biometrics against abuse, and they must also prevent fake biometrics.

We focus on fingerprint systems since they have become widespread as authentication terminals for PCs or mobile terminals. A fingerprint system has an enrollment process and a verification process. In an enrollment process, the system captures finger data from an enrollee with sensing devices, extracts features from the finger data, and then record them as a template with a personal information, e.g. a personal identification number (PIN), of the enrollee into a database. We are using the word *finger data* to mean not only features of the fingerprint but also other features of the finger, such as *live and well* features. In a verification (or identification) process, the system captures finger data from a finger with sensing devices, extracts features, verifies (or identifies) the features by comparing with templates in the database, and then outputs a result as *Acceptance* only when the features correspond to one of the templates. Most of fingerprint systems utilize *optical* or *capacitive* sensors for capturing fingerprints. These sensors detect difference between ridges and valleys of fingerprints. Optical sensors detect difference in reflection. Capacitive sensors, by contrast, detect difference in capacitance. Some systems utilize other types of sensors, such as *thermal* sensors, *ultrasonic* sensors. In this study we examine fingerprint systems which utilize optical or capacitive sensors.

Potential threats caused by something like real fingers, which are called *artificial fingers*, should be crucial for authentication based on fingerprint systems. However, vulnerability evaluation against attacks using such artificial fingers has been rarely disclosed.

As researchers who are pursuing secure systems, we would like to discuss attacks using artificial fingers and conduct experimental research to clarify the reality. We report that

1. *gummy fingers*, namely artificial fingers that are easily made of cheap and readily available gelatin, were accepted by extremely high rates by 11 particular fingerprint devices with optical or capacitive sensors [2], and
2. *conductive silicone fingers*, namely artificial fingers that are made of silicone rubber filled with electrically conductive carbon black of 12%-16%, were accepted by extremely high rates by the same set of fingerprint devices except for two devices using optical sensors with seemingly color-checking ability [3].

We have used the molds, which we made by pressing our live fingers against them, or by processing fingerprint images from prints on glass surfaces, or by processing impression of inked fingers. We describe how to make the molds, and then show that the gummy fingers and conductive silicone fingers which are made with these molds, can fool the fingerprint devices.

The fact that gummy fingers which are easy to make with cheap and easily obtainable tools and materials can be accepted suggests review not only of fingerprint systems but also of biometric systems. This experimental study on the artificial fingers will have considerable impact on security assessment of biometric systems. Manufacturers and vendors of biometric systems should carefully examine security of their system against artificial clones. Also, they should make public results of their examination, which lead users of their system to a deep understanding of the security. We would like to discuss the effect of such a vulnerability analysis and how to disclose the information based on our experience and the responses we received [4].

## References

1. Jain, K.: Introduction to biometrics, in Biometrics: Personal Identification in Networked Society, The Kluwer Academic, International Series in Engineering and Computer Science, Jain, A. K., Bolle, R. and Pankanti, S. eds., Vol. 479, Chapter 1, pp. 1-41, 1999.
2. Matsumoto, T., Matsumoto, T., Yamada, K., and Hoshino, S.: Impact of Artificial "Gummy Fingers" on Fingerprint Systems, Optical Security and Counterfeit Deterrence Techniques IV, Rudolf L. van Renesse, editor, Proceedings of SPIE Vol. 4677, SPIE – The International Society for Optical Engineering, pp.275-289, 2002.
3. Endo, Y. and Matsumoto, T.: Can we make artificial fingers that fool fingerprint systems? – Part W –, Proc. of IPSJ for Computer Security Symposium, 2002.
4. Matsumoto, T.: What will you do if you find a particular weakness of a security technology?, Journal of IEICE, Vol. 84, No.3, 2001.

# Author Index

- Abe, Masayuki 206, 415  
Asano, Tomoyuki 433
- Barkan, Elad 160  
Bellare, Mihir 397  
Biham, Eli 160, 254  
Boneh, Dan 451  
Bosma, Wieb 46  
Bresson, Emmanuel 497
- Carlet, Claude 484  
Catalano, Dario 299  
Chang, Ku-Young 143  
Chang, Yan-Cheng 110  
Chee, Seongtaek 176  
Chevassut, Olivier 497  
Courtois, Nicolas T. 267  
Cramer, Ronald 206
- Damgård, Ivan 125  
D'Arco, Paolo 346  
Dent, Alexander W. 100  
Dunkelman, Orr 254
- Fehr, Serge 206  
Fujisaki, Eiichiro 125
- Gaudry, Pierrick 311  
Gentry, Craig 548  
Golle, Philippe 451  
Gouget, Aline 484  
Granboulan, Louis 364  
Gupta, Kishan Chand 466
- Hanaoka, Goichiro 81  
Hanaoka, Yumiko 81  
Hevia, Alejandro 379  
Hong, Dowon 143  
Hong, Seokhie 243  
Hsiao, Chun-Yun 110  
Hutton, James 46
- Imai, Hideki 81
- Jakobsson, Markus 451  
Juels, Ari 451
- Jung, Seokwon 243
- Katz, Jonathan 192  
Keller, Nathan 254  
Kim, Jongsung 243  
Kim, Kwangjo 533  
Klimov, Alexander 288  
Koga, Hiroki 328  
Kumar, M.V.N. Ashwin 224  
Kurosawa, Kaoru 64
- Lee, Sangjin 243  
Lee, Wonil 243  
Lenstra, Arjen K. 1  
Lim, Jongin 176  
Lu, Chi-Jen 110  
Lucks, Stefan 27
- Matsumoto, Tsutomu 574  
Micciancio, Daniele 379  
Mityagin, Anton 288  
Moon, Dukjae 243
- Namprempre, Chanathip 515  
Neven, Gregory 397  
Nguyen, Phong Q. 299
- Ogata, Wakaha 64  
Ohkubo, Miyako 415
- Park, Sangwoo 176  
Pieprzyk, Josef 267  
Pointcheval, David 497
- Rangan, C. Pandu 224  
Ryu, Heuisu 143
- Sarkar, Palash 466  
Shamir, Adi 1, 288  
Shikata, Junji 81  
Silverberg, Alice 548  
Srinathan, K. 224  
Stern, Jacques 299  
Stinson, Douglas R. 346  
Sung, Soo Hak 176  
Suzuki, Koutarou 415

Tomlinson, Jim 1

Tromer, Eran 1

Verheul, Eric R. 46

Yoon, E-Joong 176

Yung, Moti 192, 567

Zhang, Fangguo 533

Zhong, Sheng 451